HC/3 Controller User's Manual

References found in this document to prior company and division names such as General Scanning Inc., GSI Lumonics, and GMAX now refer to General Scanning Optical Scanners, a member of GSI Group Inc.

P/N: 7OM-034
Revision G

www.gs-scanners.com

Contact Information

Americas

39 Manning Road

Billerica, MA 01821

U.S.A.

TEL: +1 (978) 439-5511

FAX: +1 (978) 663-0131

E-mail: ScannerSales-Americas@gsig.com

Toll Free: +1 (800) 342-3757

Europe

Einsteinstrasse 2

D-85716 Unterschleissheim

Germany

TEL: +49 (89) 31707-0

FAX: +49 (89) 31707-250

E-mail: ScannerSales-Europe@gsig.com

Asia

Technoport Kamata, 16-1

Minami-Kamata 2-Chome,

Ohta-Ku Tokyo 144-0035, Japan

TEL: +81 (3) 5425-7733 [Sales]

+81 (3) 5714-0557 [Service]

FAX: +81 (3) 5425-7738 [Sales]

+81 (3) 5714-0566 [Service]

E-mail: ScannerSales-Asia@gsig.com

Table of Contents

1 IMPORTANT INFORMATION

1.1 ESD WARNING

The OEM electronics that *General Scanning* manufactures - including galvanometers and servo controllers - are electrostatic discharge (ESD) sensitive. Improper handling could therefore damage these electronics. *General Scanning* has implemented procedures and precautions for handling these devices and we encourage our customers to do the same. Upon receiving your components, you should note that it is packaged in an ESD-protected container with the appropriate ESD warning labels. The equipment should remain sealed until the user is located at a proper static control station*.

Note: Any equipment returned to the factory must be shipped in anti-static packaging.

(*) A proper static control station **should** include:

1. A soft grounded conductive tabletop or grounded conductive mat on the tabletop.
2. A grounded wrist strap with the appropriate (1 Meg) series resistor connected to the tabletop mat and ground.
3. An adequate earth ground connection such as a water pipe or AC ground.
4. Conductive bags, trays, totes, racks or other containers used for storage.
5. Properly grounded power tools.
6. Personnel handling ESD items should wear ESD protective garments and ground straps.

1.2 Warranty Information

The Customer shall examine each shipment within 10 days of receipt and inform General Scanning of any shortage or damage. If no discrepancies are reported, General Scanning shall assume the shipment was delivered complete and defect free. General Scanning warrantees products against defects up to 1 year from manufacture date, barring unauthorized modifications or misuse. Repaired product is warranted 90 days after the repair is made, or one year after manufacture date - whichever is longer.
Contact Customer Service to obtain a Return Materials Authorization number *before returning any product for repair*.
All orders are subject to the General Scanning Terms and Conditions and Limited Warranty. Visit our website for the latest version of these documents and other useful information.

IMPORTANT: Optical Scanners are normally tuned, serialized and warranted as a matched set for optimized performance. Mismatched components negatively affect performance and void the warranty. A matched set typically consists of galvanometer motor, mirror load, electronic driver board and interface cable.

Customers assume all responsibility for maintaining a laser-safe working environment. OEM customers must assume all responsibility for CDRH (Center for Devices and Radiological Health) certification.

1.3 Customer Support

General Scanning has support services to address your questions or concerns with either the product or manual you are using. Before calling for assistance, be sure to refer to any appropriate sections in the manual that may answer your questions. Call General Scanning's Customer Service Department Monday through Friday between 8 A.M. and 5 P.M. local time (GMT –05:00 Eastern Time (US & Canada)).

The customer service personnel will be able to give you direct assistance and answers to your questions.

1.4 Unpacking

Carefully unpack the contents from the box and inspect each item for damage. Check the contents of the box against the packing list to ensure reception of all parts. Contact customer service immediately if you suspect shipping damage or an incomplete shipment. Save the shipping container and packaging material in case you need to return unit for service.

2 INTRODUCTION

General Scanning continues the Helpercard line of scan and laser control products with the HC3 card. The HC3 fits into any PCI-bus offered standard with most common desktops to provide XY2-100 communication to digital scan heads. The new release includes greater functionality including CO_2 tickle pulse, first pulse suppression, and option-isolation of IO ports. The IO2 add-on card allows additional analog control required for some lasers. General Scanning offers WinMCL Plus to communicate to the HC3. WinMCL Plus functions on Windows NT, 2000, or XP. The HC3 can be configured with PC MarktMT to backward compatible with PCMarkMT MS DOS based systems.

Throughout this document are blue hot-links for easy navigation. In addition, headings in the table of contents can be clicked for rapid movement to the selected topic.

2.1 General Description

Optical scanning systems consist of three distinct interdependent items. First, the XY scan head includes the galvos, mirrors, and applicable two-axis mounting assembly. Second, closed loop scanners employ high accuracy servos that adjust the mirror position based on position feedback sensing and command input signals. And third, a controller device supplies a command to the servo determining the position and velocity of the system. The galvanometer/mirror scan head, servos, and controller execute in synergy to optimize the most accurate and fastest performance possible.

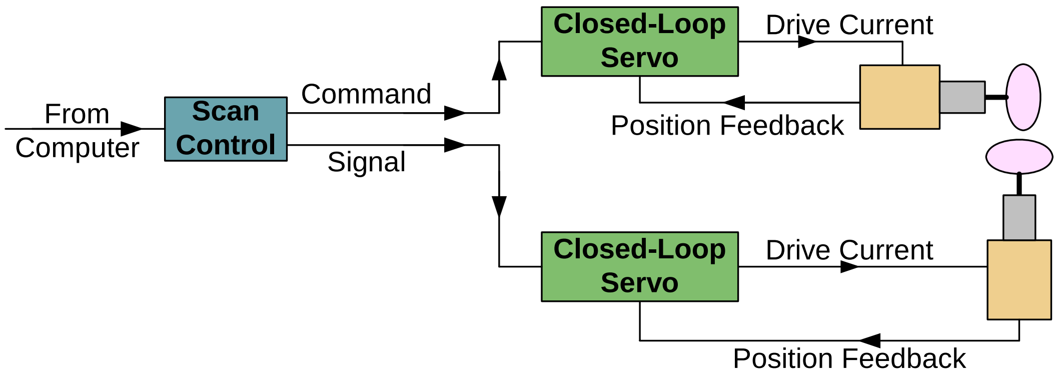


Figure 1: Typical scheme of a two-axis optical scanning system, including galvos, servos, and control electronics.

2.2 HC3 Feature Set

2.2.1 PCI Interface

- 32 bit 33 MHz PCI host interface supporting 132 Mbytes/sec burst transfers
- Complies with PCI v2.2 specification
- Plug & Play compatibility
- 3.3V or 5V PCI signaling

2.2.2 Software Compatibility (requires PCI driver)

- MMCL and higher layers (PCMarkMT, Job Editor)
- Win MCL Plus and higher layers

2.2.3 HC3 Functionality

- Serial link interface
- Optically isolated laser control I/O
- 50-pin internal I/O compatibility
- Master/slave compatibility
- Interface to General Scanning two-axis XY scan heads
- Interface to General Scanning three-axis XYZ scan systems
- One input data channel for the status of the receivers
- First in first out (FIFO) ram
- Shutter control circuitry
- Interrupt control logic

2.2.4 On-Board IO2 Functionality

The HC3 includes most functionality the HC2 required the IO2 add-on card to accomplish.

- BEGIN_MARK and STOP_MARK differential optically isolated inputs
- Single-ended optically isolated output for First Pulse Suppression.
- Optional LM inhibit on Laser Error

2.2.5 Enhancements

- Configurable on-board instruction store (up to 64K 32-bit words)
- Support for low-latency binning
- Configurable CO_2 tickle pulse generation
- Configurable First Pulse Suppression generation
- Shutter relay control
- Inhibit galvo movement on mark abort
- Inhibit laser firing on start-up
- Short step periods (10µS), therefore smaller microsteps, and better quality for certain sensitive applications

3 INSTALLATION & CONFIGURATION

THe HC2 card requires adjustment of various settings to configure the board for best application performance. The following section outlines the installation and setup of the scan card into a computer.

3.1 Minimum Computer Requirements

In general, as long as the selected computer can run the Windows operating system, then the computer will also be capable of running the HC3 card and WinMCL Plus.

Computer Minimal Configuration	
Processor	Pentium 100MHz with a PCI bus
Operating System	Windows NT, 2000, or XP; MS DOS
RAM	64MB recommended
Graphic Card	VGA (16-bit)
Monitor	VGA – Color
Mouse	Microsoft mouse or compatible
Extension Slot	Minimum one PCI bus slot
Hard Disk	Minimum 5MB free
Floppy Disk	3.5" / 1.44MB CD-ROM Drive

3.2 Card Installation

Be sure to set any jumper settings before installing the card. Do the following to install the HC3 into the computer:

1. Be sure to set power and laser modulation jumpers W1 and W2 before installing the card. See section 0 Opto-Isolation and section 0 Laser Modulation Signal for more details.
2. With the computer and the monitor power off, remove the computer's cover.
3. Select a PCI slot on the motherboard. Remove the existing slot blank and retaining screw.
4. Firmly insert the HC3 into the slot. Verify that the 9 and 25 pin connectors will face outward of the PC cover when the cover is reinstalled.
5. Secure the board to the computer with the retaining screw. Make sure the HC3 is firmly seated in the computer and the computer frame is square.
6. Replace and secure the computer cover.

After the card is installed the hardware device found wizard in Windows plug-and-play machines will appear. Hit cancel and proceed with the WinMCL Plus installation.

3.3 HC3 LED Definitions

The HC3 includes several LEDs, located at the top edge of the card, denoting the state of the HC3 card. The LED lights serve as a useful troubleshooting tool. See 0 Location of Jumpers and LEDs for the location of each LED.

LED	Description
D1	Start Mark asserted
D2	Stop Mark asserted
D3	/LM asserted (default polarity)
D4	Host access in progress
D5	Shutter relay closed
D6	Mark in progress

3.4 Interconnections & Signal Definitions

The HC3 card includes three separate signal locations: 25-pin XY2-100 serial link connector for scan head control, 9-pin opto-isolated laser and flag IO, and 50-pin non opto-isolated laser and flag IO header. Several of the signals on the 9-pin connector are also available without opto-isolation on the 50-pin header. Figure 2 depicts a typical scanning configuration.

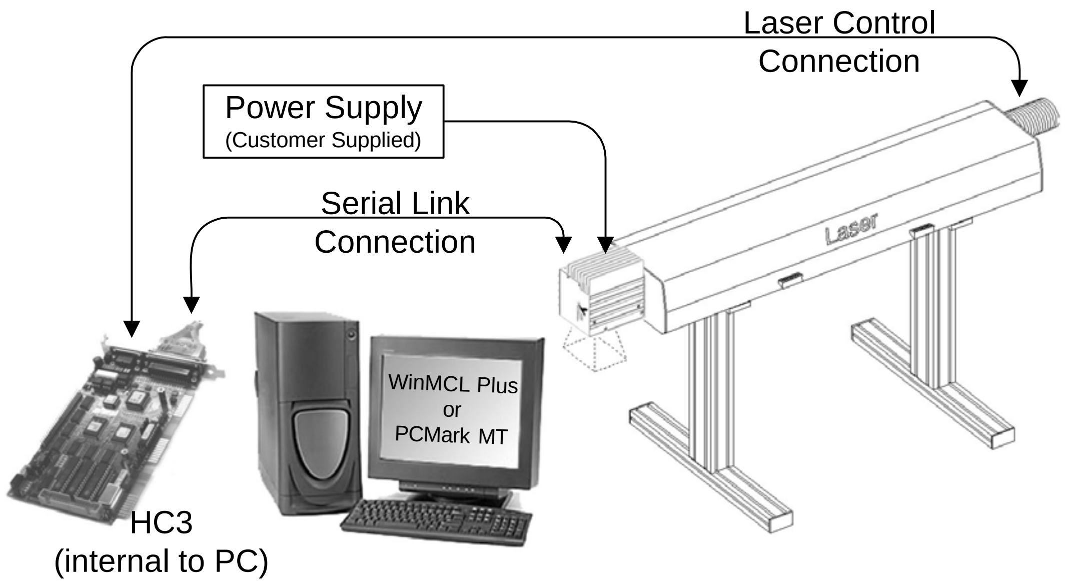
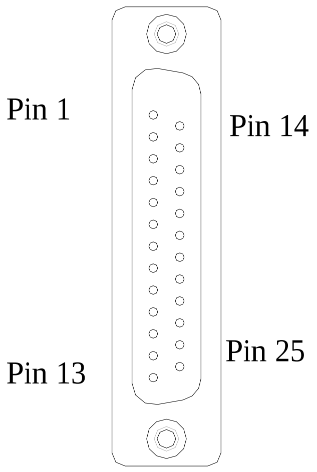


Figure 2: Typical two-axis laser scanning interconnection setup.

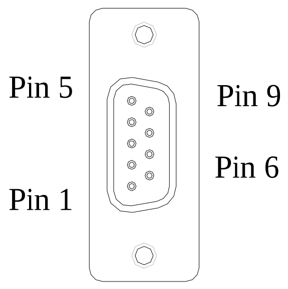
3.4.1 Serial Link Connector

The HC3 relays standard XY2-100 protocol to the digital interface head through a digital control cable, General Scanning stocks standard cable lengths of 3 and 10 meters. The following chart shows the 25-pin digital output to the scan head for reference only.

INTERFACE	PIN	ASSIGNMENT
 25-Pin D-sub (Female) Digital I/O Connection		***HC3 Connections***
	1	SENDCK -
	14	SENDCK +
	2	SYNC -
	15	SYNC +
	3	CHANNEL X -
	16	CHANNEL X +
	4	CHANNEL Y -
	17	CHANNEL Y +
	5	RESERVED FOR Z AXIS
	18	RESERVED FOR Z AXIS
	6	STATUS -
	19	STATUS +
	7	NOT CONNECTED
	20	NOT CONNECTED
	8	RESERVED
	21	RESERVED
	9	LM-
	22	LM+
	10	NOT CONNECTED
	23	SHIELD GROUND
	11	SHIELD GROUND
	24	SHIELD GROUND
	12	NOT CONNECTED
	25	SHUTTER OUT
	13	SHUTTER IN

3.4.2 External IO 9-pin Connector

The 9-pin connector external to the PC provides opto-isolated signals for various inputs and outputs. Laser Modulation (LM), Laser Error input, Mark Abort input, Begin Mark input, and Mark In Progress output are available on the internal 50-pin header without opto-isolation. GROUND$_{OPTO}$ and VCC$_{OPTO}$ are the same as GND and V5_0 found on the 50-pin header when W1 is installed. If W1 not installed, the 9-pin input and output signals require external $^{+}$5V power.

INTERFACE	**PIN**	**ASSIGNMENT**
 9-Pin D-sub (Male) Power Connection	1 6 2 7 3 8 4 9 5	GROUND$_{OPTO}$ VCC$_{OPTO}$ (+5V) LM_OUT / FLAG_INPUT (INPUT) / STOP_MARK (INPUT) / REMOTE_EXECUTE (OUTPUT) / START_MARK (INPUT) / MARK_ERROR (OUTPUT) / MARK_IN_PROGRESS (OUTPUT)

External IO Pin Definitions	
Pin 1	GROUND$_{OPTO}$ = Input or output defined by jumper W1.
Pin 2	LM_OUT provides laser modulation. Polarity defined by jumper W2.
Pin 3	Connects to the OPTO isolator O8. It is set as an input /STOP_MARK.
Pin 4	Connects to the OPTO isolator O8. It is set as an input /START_MARK.
Pin 5	Connects to the OPTO isolator O1. It is set as an output /MARK_IN_PROGRESS.
Pin 6	VCC$_{OPTO}$ = +5V input or output defined by jumper W1.
Pin 7	Connects to the OPTO isolator O9. It is set as an input /FLAG_INPUT.
Pin 8	Connects to the OPTO isolator O2. It is set as an output /REMOTE_EXECUTE.
Pin 9	Connects to the OPTO isolator O2. It is set as an output /MARK_ERROR.

External 9-pin signals also available on internal 50-pin header		
Signal	**9-Pin Connecter**	**50-Pin Header**
LM_OUT	Pin 2	Pin 33
/ STOP_MARK	Pin 3	Pin 20
/ START_MARK	Pin 4	Pin 17
/ MARK_IN_PROGRESS	Pin 5	Pin 21
/ FLAG_INPUT	Pin 7	Pin 19

Signals cannot be opto-isolated on 50-pin header.

Location of Jumpers and LEDs

The following figure shows the location of jumpers W1 and W2 on the HC3 card. See sections 3.3 HC3 LED Definitions, 0 Opto-Isolation, and 0 Laser Modulation Signal for more information.

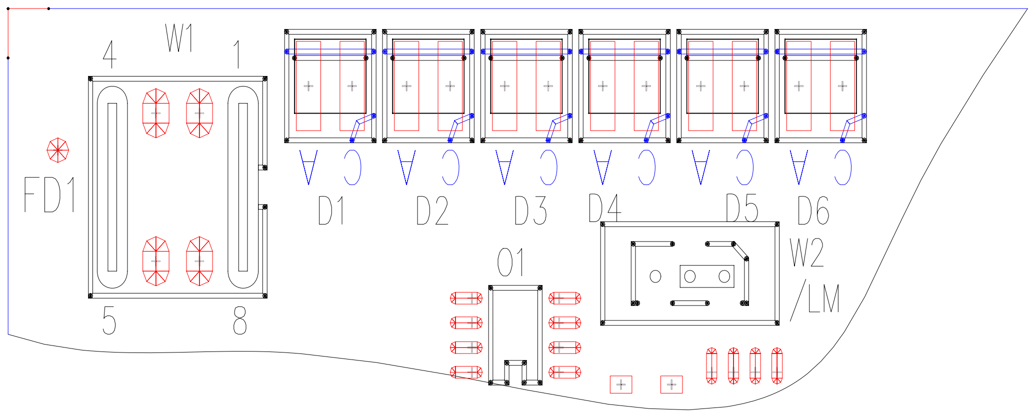


Figure 3: Location of HC3 jumpers and LEDs.

Opto-Isolation (Jumper W1)

The output logic can be supplied from the internal PC $^+$5V supply or from an external power supply.

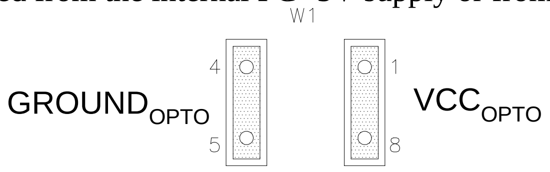


Figure 4: W1 pin configuration.

The default configuration bridges a jumper across pins 1 to 8 and pins 4 to 5 setting the HC3 to use the internal $^+$5V supply of the computer. To achieve opto-isolation, remove the jumper and connect an external power supply to the external 9-pin connector. The input voltage is defined as:

VCC$_{OPTO}$ = 4.75 to 5.75V max. 100mA	W1, Pin 8
GROUNDOPTO = VCC$_{OPTO}$ Return	W1, Pin 5

Laser Modulation (LM) Signal (Jumper W2)

The LM signal is active high or active low depending on the position of jumper W2. For /LM = low active, insert W2 between pin 1 & 2 (default setting). For LM = high active, insert W2 between pin 2 & 3. The jumper setting does not determine the output of laser modulation on the 50-pin header.

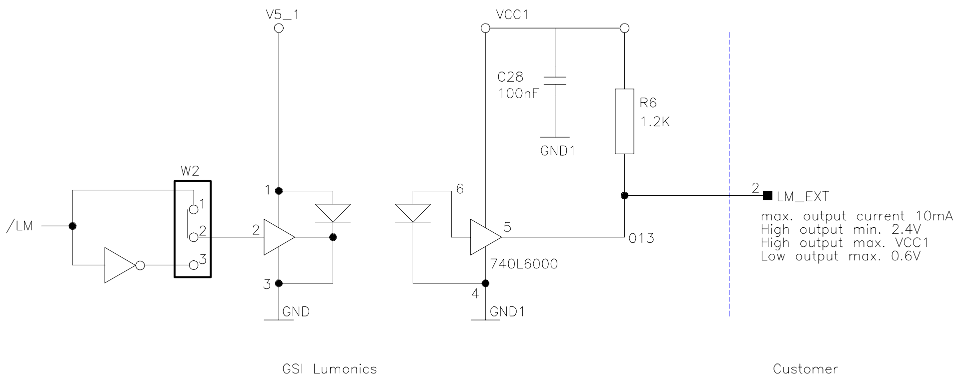


Figure 5: HC3 laser modulation schematic

Input Flag Signals

The following diagram shows the interface circuitry for input signals /START_MARK, /STOP_MARK, and /FLAG_INPUT. /FLAG_INPUT is also sometimes referred to as /FLAG_LER on the 50-pin header or /LASER_ERROR. The flag inputs require minimum 0.3mA to maximum 10mA at 5Volts.

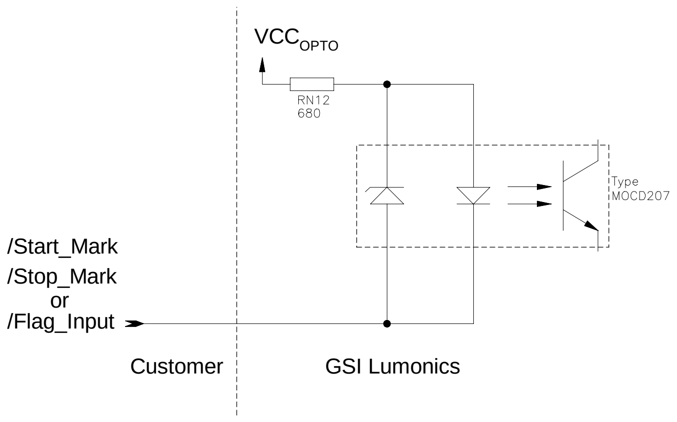


Figure 6: HC3 schematic for all input flags.

Output Flag Signals

The following diagram shows the interface circuitry for output signals /REMOTE_EXECUTE, /MARK_ERROR, and /MARK_IN_PROGRESS. All the flags are low active. Be aware that the flag outs are low current devices.

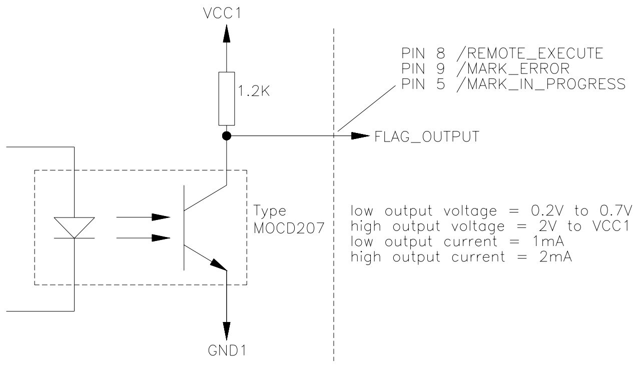


Figure 7: HC3 schematic for all output flags.

Fuse for External IO Circuitry

A resettable fuse is fitted to the HC3 to protect the external IO circuitry against incorrect connections (e.g. wrong polarity or exceeding the maximum input voltage). The fuse is labeled "F1" on the HC3 card.

3.5 Internal IO 50-pin Header

The internal 50-pin header provides laser signal access without opto-isolation. All signals are TTL outputs with a maximum 10mA low current. /REMOTE_EXECUTE and /MARK_ERROR are not available on the 50-pin head and can only be found on the external 9-pin header. The 50-pin header features signals regarding:

- Laser modulation (LM)
- First plus suppression trigger pulse
- Remote power control flag, /LP_COUT
- Relay contact for shutter circuitry

A short on a signal line will destroy the HC3 immediately!

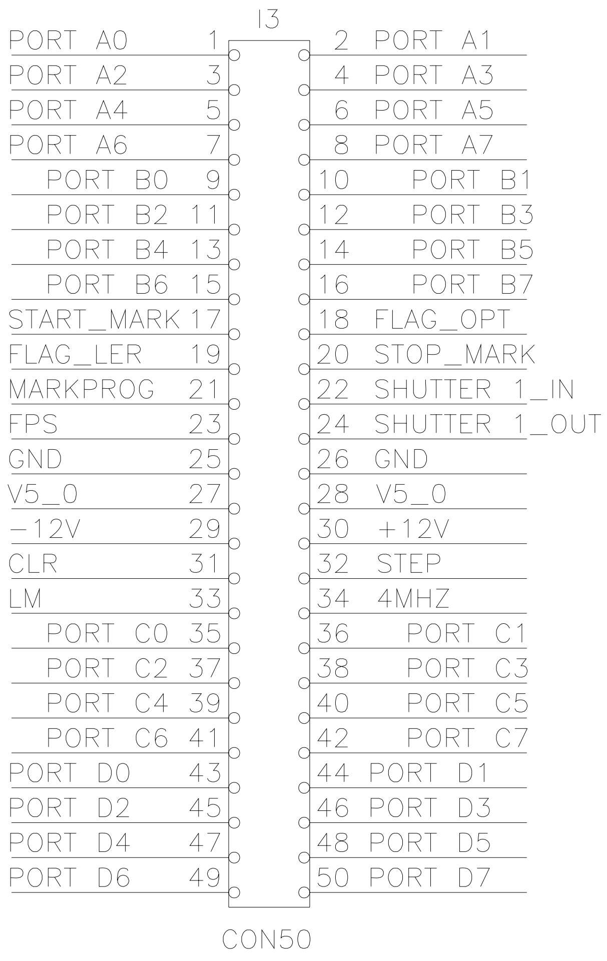
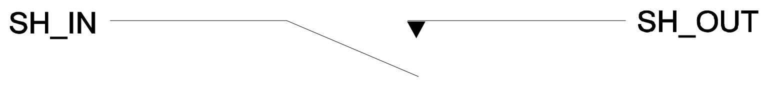


Figure 8: Pin assignment of the internal IO header.

Laser Safety Shutter

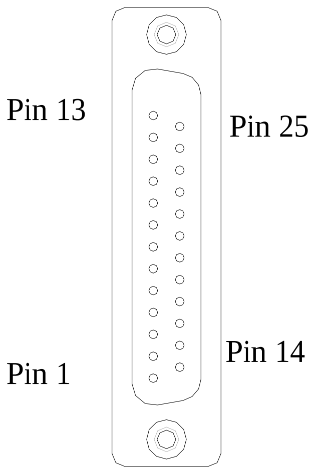
The use of the laser safety shutter feature should replace the "Safety and Warnings" referenced in all General Scanning scan head manuals or the user site laser safety policies.

To improve laser safety, the shutter should be connected in a series with SH_IN and SH_OUT. The circuit enables the shutter control line only when the computer is switched on. Contact ratings are maximum 48V, maximum 0.5A, and maximum switch on resistance 200mΩ.



50-Pin Header Adaptor Cable

General Scanning offers an adaptor cable for the internal 50-pin header, General Scanning part number 712-78735. Contact customer service regarding information on ordering the cable.

INTERFACE	PIN	ASSIGNMENT
 25-Pin D-sub (Male) Digital IO Connection		***HC3 Connections***
	1	PORT B0
	2	PORT B1
	3	PORT B2
	4	PORT B3
	5	PORT B4
	6	PORT B5
	7	PORT B6
	8	MSB (/LP_COUT)
	9	PORT N15
	10	FPS TRIGGER
	11	SHUTTER IN
	12	GROUND
	13	+5V
	14	GROUND
	15	PORT N14
	16	+5V
	17	PORT N16
	18	MIP
	19	PORT A3
	20	PORT A2
	21	PORT A1
	22	+5V
	23	SHUTTER OUT
	24	LASER MODULATION
	25	PORT A0

3.6 Configurable Logic

The HC3 contains on-board configurable logic for adjusting hardware settings regarding laser modulation, CO_2 tickle pulse, and first pulse suppression. The settings can be configured from Windows using WinMCL Plus. If working with PCMarkMT, then an additional file named HC3Setup.exe that comes with the software will be required to set the following features.

3.6.1 CO2 Tickle Pulse Generation

The HC3 board can be optionally configured through the PCI 9054 configuration EEPROM to provide a PWM signal on the /LM output specifically designed for CO_2 lasers. When this configuration is set, the /LM signal generated by the mark control logic is modulated by a fixed PWM frequency (either 5 kHz or 20 kHz). When the /LM signal generated by the mark control logic is de-asserted, a “tickle pulse” is imposed onto the /LM output to keep the CO_2 laser just below the lasing energy level. The tickle pulse has a fixed 5 kHz frequency with a configurable pulse width of 0.25, 0.50, 0.75, 1.00, 1.25, 1.50, 1.75, or 2.00 μSec.

3.6.2 First Pulse Suppression

/FPS is a status output available only on the internal 50-pin IO interface. In the standard configuration, first pulse suppression (FPS) is implemented as a short output pulse of 625nSec. An abort condition will not affect the FPS activation state; however, a board reset will deactivate it.

The HC3 board can be optionally configured to extend the /FPS output and alter it's polarity. Depending on the configuration setting, /FPS can be extended until the first through the seventh rising edge of the /LM output after /FPS is initially asserted, and the polarity of the /FPS output can be changed to positive assertion.

3.6.3 HC3Setup.exe (MS DOS only)

WinMCL Plus includes functionality to adjust the following settings, but PCMarkMT requires the execution of a separate program to configure the HC3 configuration logic. In the PCMarkMT tree, find the file named hc3setup.exe in directory C:\mark\tools. The hc3setup files allows the user to set and read the HC3 parameters. Type **hc3setup** and the screen will render the current parameters.

```
HC3 Parameters
Number of cards is 1 (Unless in Master/Slave mode)
       1   2   3   4
lm=    0   0   0   0    Inhibit LM on /FLAG_INPUT   (1=Inhibit)
tc=    0   0   0   0    Enable CO2 tickle pulse     (1=Positive)
tw=    0   0   0   0    CO2 PWM tickle width        (nx.25 microsec n=0-7)
fd=    0   0   0   0    /FPS pulse extension        (0-7) (up to seven LM cycles)
fi=    0   0   0   0    /FPS signal polarity        (1=positive)
```

Example: Type the following to change the parameters:

```
C:\mark\tools>hc3setup fi:1 tc:1 tw:6
```

You must reboot the computer for the configuration changes to take effect.

Now, type **hc3setup** and the screen will identify:

```
HC3 Parameters
Number of cards is 1 (Unless in Master/Slave mode)
       1   2   3   4
lm=    0   0   0   0    Inhibit LM on /FLAG_INPUT   (1=Inhibit)
tc=    1   0   0   0    Enable CO2 tickle pulse     (1=Positive)
tw=    6   0   0   0    CO2 PWM tickle width        (nx.25 microsec n=0-7)
fd=    0   0   0   0    /FPS pulse extension        (0-7) (up to seven LM cycles)
fi=    1   0   0   0    /FPS signal polarity        (1=positive)
```

4 Safety and Warnings

The United States Food and Drug Administration, through the Center for Devices and Radiological Health (CDRH), has promulgated regulations (21 CFR parts 1000 and 1040) controlling the safety of lasers and laser products for sale or manufacture in the United States.

This section is a guide to the specific areas of this product where laser safety should be addressed. General Scanning XY Scan heads are designed to provide maximum flexibility and ease of use. Such a design inherently requires the user to assure the overall safety of the configuration in use.

Note: Prior to operating any configuration of the General Scanning XY Scan heads, you must make a thorough analysis of system safety. Key information for this purpose is contained in this manual. You should become familiar with all this information before proceeding.

A full description of laser hazard analysis is beyond the scope of this manual. A technical survey of laser safety requirements can be found in **ANSI Z136.1**, **"American National Standard For the Safe Use of Lasers"**. This is available from:

American National Standards Institute, Inc.
1430 Broadway
New York, New York 10018
www.ansi.org

Among the many other sources of laser safety information, the following institution offers several excellent publications:

The Laser Institute of America
5151 Monroe Street, Suite 118W
Toledo, Ohio 43623
www.laserinstitute.org

Your Laser Safety Officer or a competent specialist in this field should make final analysis of all safety features. The first consideration in a safety analysis is the laser mated to the General Scanning XY Scan heads. The Laser Class label on the device indicates the approximate hazard level of the laser. Refer to **ANSI Z136.1** for definitions of laser classes and labeling information. Note that, besides radiation, lasers may present other hazards, e.g. electric shock or creation of poisonous fumes.

Note: The General Scanning XY Scan heads provide you with the ability to aim the laser beam over a roughly pyramidal volume. The divergence of the focused beam beyond the focal point, which is a function of the lenses selected and their position, can cause radiation to exit the pyramid. When analyzing safety, you must consider all regions within this aiming pyramid, the divergent beam, and the effects of all focal possibilities in the zone of hazard. Reflections must also be considered.

APPENDIX A: ADDITIONAL RESOURCES

GSI Lumonics supplies the following manuals for more information regarding related components. Visit the General Scanning Component Group's website to download any of the additional manuals listed below, or contact technical services with a manual request or any further questions.

Galvanometers:

VM500 & VM1000 User Manual: GSIL Part #7OM-028

VM2000 User Manual: GSIL Part #7OM-022

Servo Control:

MiniSAX User Manual: GSIL Part #176-25016

ISD Hardware Manual: GSIL Part #7OM-031

Scan Heads:

HPLK Hardware Manual: GSIL Part #7OM-1020

HSM15M2 Hardware Manual: GSIL Part #7OM-1060

HBX10 Hardware Manual: GSIL Part #7OM-1015

Software Options:

PC-MARK MT™/ PC-MARK Command Reference: GSIL Part #176-25008

PC-MARK MT Programmer's Manual: GSIL Part #176-25015

ScribeSmart™ User Manual: GSIL Part #7OM-1117

WinMCL Plus Technical Reference: GSIL Part #7OM-1095

Postgrid User Manual: GSIL Part #176-25005

APPENDIX B: HC/3 BLOCK DIAGRAM

Four major functional blocks are incorporated in the HC3: the PCI (host) interface, the SRAM, the programmable logic, and the IO (application) interfaced.

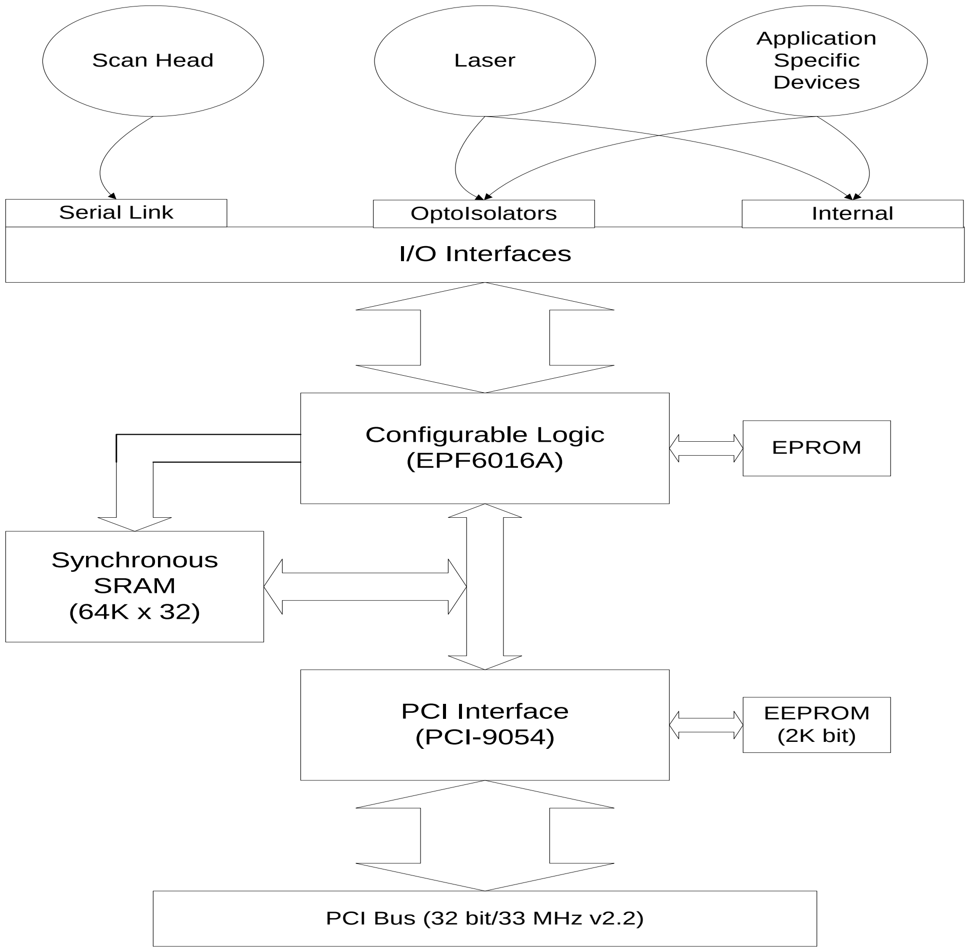

Scan Head
Laser
Application Specific Devices
Serial Link
OptoIsolators
Internal
I/O Interfaces
Configurable Logic (EPF6016A)
EPROM
Synchronous SRAM (64K x 32)
PCI Interface (PCI-9054)
EEPROM (2K bit)
PCI Bus (32 bit/33 MHz v2.2)

Figure 9: HC3 block diagram

APPENDIX C: REGULATORY

The HC3 board, when incorporated into a laser marking system, will meet the following requirements of the EMC Directive of the EEC (89/336/EEC/93/68/EEC) necessary for a CE declaration of conformity and CE marking:

EN55011: 1991	RF Emissions (from industrial equipment)
EN61000-3-2: 199	AC Power Mains Harmonic Current Emissions
EN61000-3-3: 1995	AC Voltage Fluctuations and Flicker
EN61000-4-2: 1995	Electrostatic Discharges
EN61000-4-3: 1996	Immunity to Radiated RF Electromagnetic Fields
EN61000-4-4: 1995	Electrical Fast Transient/Burst
EN61000-4-5: 1995	Surge Immunity
EN61000-4-6: 1996	Conducted Disturbance Induced by RF above 9KHz
EN61000-4-11: 1994	Mains Voltage Variations

APPENDIX D: HC/3 MASTER SLAVE

This version of the HC3 card allows up to four interconnected HC3 M-S cards in one PC to control up to four General Scanning two-axis or three-axis laser systems simultaneously. Each scan head laser system needs a corresponding HC3 M-S. One of the cards serves as a master and will control the first head, the laser modulation, and the timing while the other cards serve as slaves and each control one additional head. Each system will be associated with its own calibration files through software, thus providing all systems will execute the exact same pattern.

A dual-head configuration does not require one HC3 Master-Slave card for each head. Some configurations utilize two scan heads mounted on one laser. If identical patters run on each head, then a single HC3 card may be used to control the laser firing and scan head control. In such a configuration, only one correction file can be set for both heads. Simply use a Y-connector off the 25-pin serial link of the HC3 and run a serial cable to each of the scan heads. Separate laser patterns or separate laser control cannot be run using the same HC3 card! Two lasers require their own HC3 M-S card.

The installed HC3 cards must be connected via the General Scanning cable (part number 712-777341) through the connector J3. The master end of the cable is the connector with only two wires.

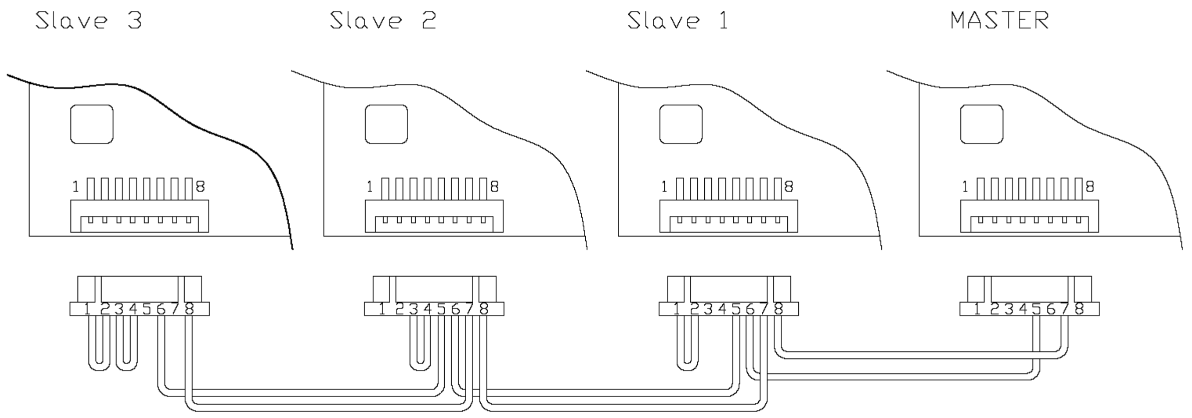


Figure 10: HC3 Master-slave cable configuration.

APPENDIX E: OPTIONAL IO-2 CARD

The IO2 add-on board contains additional options for the HC3. Typically, lasers requiring analog voltage control necessitate the IO2 addition to the HC3. The IO2 features include:

Function	**HC2**	**HC2 – IO2**	**HC3**	**HC3-IO2**
Laser Modulation output opto-isolated	✓	✓	✓	✓
Higher allowable external voltage for opto-isolation ($^+$5V to $^+$18V)		✓		✓
All outputs (on 9-pin con.) opto-isolated	✓	✓	✓	✓
7-bit D/A converter for analog power control (0V to user defined maximum between 0.7 - 10V)		✓		✓
Gate signal for first pulse suppression		✓	✓	✓
Jumper selectable 1st, 2nd, 3rd, or 4th pulse suppression		✓		✓
Software configurable 1st, 2nd, 3rd, or 4th pulse suppression		✓	✓	✓
Additional user programmable 8-bit opto-isolated input (e.g. for system flags.		✓	✓*	✓

* Not opto-isolated

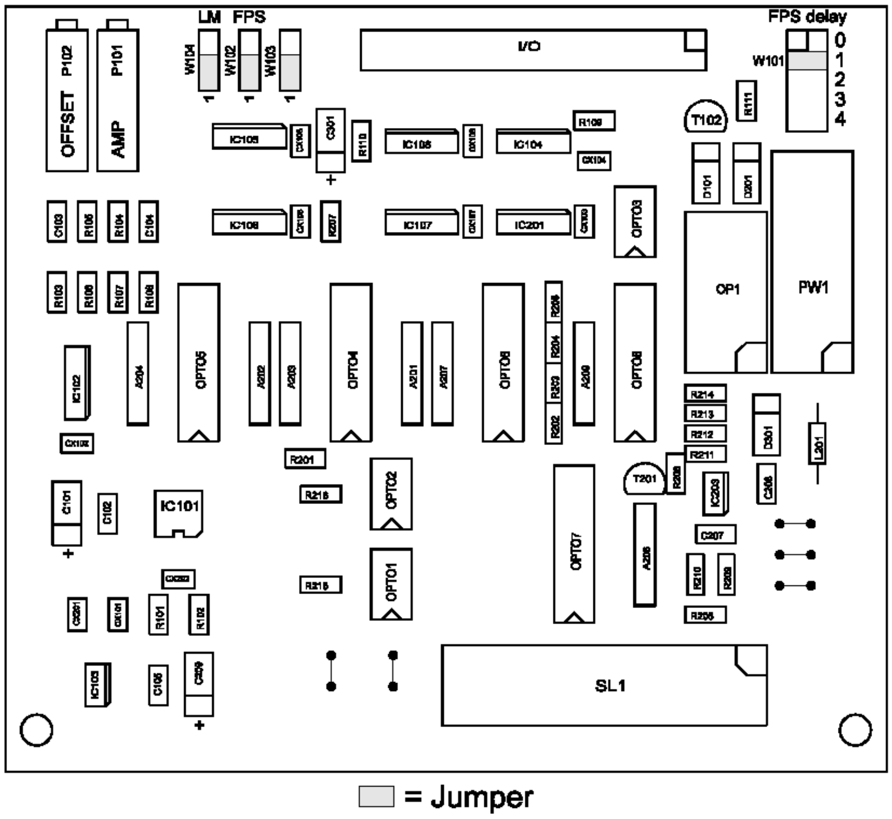


Figure 11: IO2 card outline drawing.

IO2 Card Installation

The IO2 card piggybacks onto the HC3. The HC3 must be removed from the computer, the IO2 card plugged into the HC3, the jumpers on the IO2 card set, and the card assembly reinstalled into the computer. Refer to sections 0 and 0 for information on jumper settings.

To remove the HC3:

1. With the computer and the monitor power off, remove the computer cover.
2. Remove the cables from the HC3 output connectors.
3. Unplug and remove the HC3 from the computer motherboard.

Connecting the IO2 card to the HC3:
The connector on the solder side of the IO2 card mates to the internal 50-pin IO header on the HC3. Make sure Pin1 of the IO2 card is connected to Pin1 of the HC3.

> The male header on the HC3 is a 50-pin; the female connector on the IO2 is a 34-pin connector. The higher number pins on the 50-pin HC3 connector remain unused.

1. Place screws (M3x6 mm) into the two holes on the IO2 card such that the screw heads are on the component side of the board.
2. Insert spacers over the screws.
3. Align the connector with the IO header on the HC3 and press them together.
4. Secure the each panel by placing a flat washer, lock washes, and then nut on the solder side of the HC3.

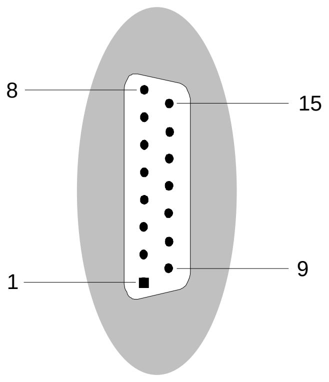
Reinstalling the card assembly:
To install the card assembly you will need two adjacent blank slots. One slot will house the HC3. The second slot will mount the two connectors on the ends of the ribbon cable coming from the IO2 card.

1. Firmly insert the card assembly into one slot.
2. Secure the HC3 to the computer with the retaining screw.
3. Remove the slot blank from the adjacent slot.
4. Secure the connector assembly to the computer with the retaining screw.
5. Replace and secure the computer cover.

IO2 Interconnections

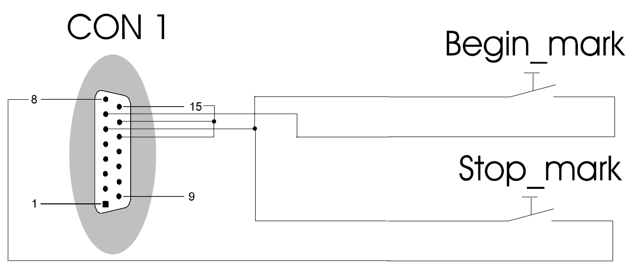
The serial link cable connecting to the scan head still interfaces from the HC3 card. With the addition of the IO2 card, two more connections exit to mating system equipment.

Connector 1 (15 pin D-sub connector)

INTERFACE CON1	PIN	ASSIGNMENT	NOTES
 15 Pin D-Sub male connector	1	$^+$VSS1	External input ($^+$5 to $^+$18 V)
	2	RESERVED	
	3	LP	Lamp current (0 to 10 V)
	4	/LP_COUT	0 V = Remote current control
	5	SH_IN	Shutter relay contact
	6	$^+$5 V	$^+$5 V output
	7	BEGIN_MARK	
	8	STOP_MARK	
	9	GROUND1	External input return
	10	LP_RETURN	Signal Ground output
	11	LOCAL	0 V = Remote control
	12	SH_OUT	Shutter relay contact
	13	GROUND	Signal Ground output
	14	/BEGIN_MARK	
	15	/STOP_MARK	

Using STOP_MARK and BEGIN_MARK

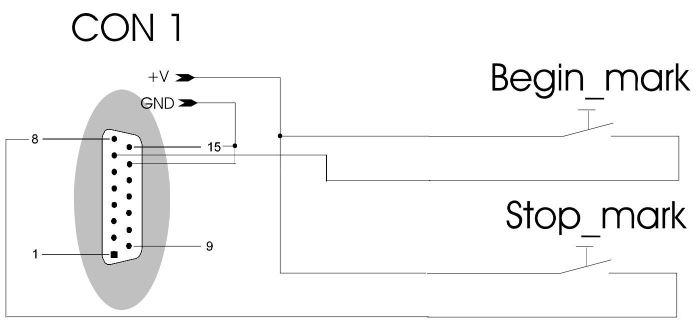
Without Opto-Isolation



- Connect Con 1 Pin 13 (ground output) to Pin 14 & Pin 15.
- Connect Con 1 Pin 6 ($^+$5V output) to the switches for STOP_MARK and BEGIN_MARK.
- Connect Con 1 Pin 8 to the return line of the STOP_MARK switch.
- Connect Con 1 Pin 7 to the return line of the BEGIN_MARK switch.

There is no extra external voltage needed if only the STOP_MARK and BEGIN_MARK function of the customer interface CON1 is used.

With Opto-Isolation



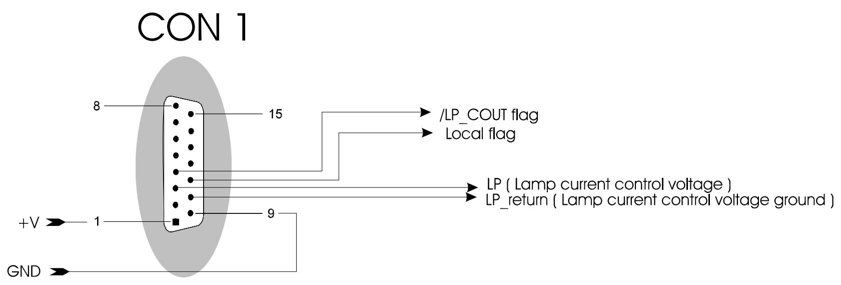
- Connect Con 1 Pin 14 & Pin 15 with the external voltage ground .
- Connect the external voltage (+5V to +18V) to the switches for BEGIN_MARK and STOP_MARK.
- Connect Con 1 Pin 8 to the return line of the STOP_MARK switch connect Con 1 Pin 7 to the return line of the BEGIN_MARK switch.

Flags and lamp current

There are two Flag outputs:

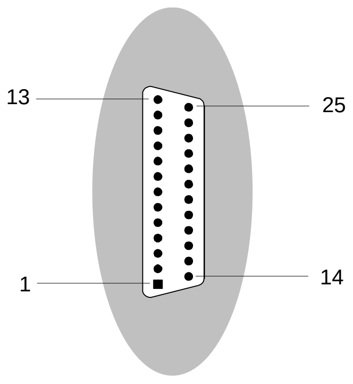
- Remote current control (/LP_COUT)
- Remote control (LOCAL)

Interconnection of the customer interface:


CON 1
8
15
/LP_COUT flag
Local flag
LP (Lamp current control voltage)
LP_return (Lamp current control voltage ground)
+V
1
9
GND

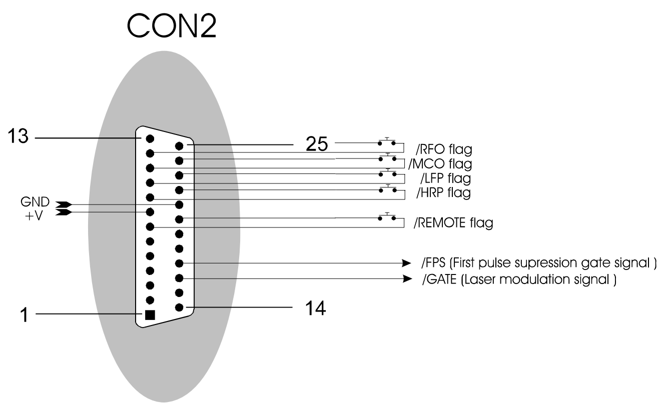
The external voltage has to be supplied to the CON1 connector if the flags and/or the lamp current control voltage are to be used.

Connector 2 (25 pin D-sub connector)

INTERFACE CON2	PIN	ASSIGNMENT	NOTES
 25 Pin D-Sub male connector	1	+VCC2	Pin 1,2,3,4,5 are connected together.
	2	+VCC2	
	3	+VCC2	
	4	+VCC2	
	5	+VCC2	
	6	NOT UESD	
	7	/REMOTE	Remote flag
	8	+VSS2	External input (+5 to +18 V)
	9	/HRP	High reverse power flag
	10	/LFP	Low forward power flag
	11	/MCO	Modulator crystal over temp.
	12	/RFO	RF modulator over temp.
	13	GROUND	
	14	XA (+VCC2)	Pin 14 and Pin 15 are for Mode GATED-CW
	15	XB (GROUND)	
	16	/GATE	Laser Modulation
	17	/FPS	First pulse suppression
	18	GROUND	
	19	NOT USED	
	20	/REMOTE RETURN	Signal Ground
	21	GROUND	External Ground input
	22	/HRP RETURN	Signal Ground
	23	/LFP RETURN	Signal Ground
	24	/MCO RETURN	Signal Ground
	25	/RFO RETURN	Signal Ground

HF-driver interface

The HF-driver interface is designed for Q-switched YAG laser systems (PIN configuration for reference only[1]).

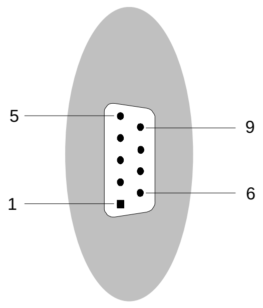

CON2
13
25
/RFO flag
/MCO flag
/LFP flag
/HRP flag
GND
+V
/REMOTE flag
/FPS (First pulse supression gate signal)
/GATE (Laser modulation signal)
1
14

The external voltage is normally between $^{+}5$V to $^{+}18$V.

[1] e.g. Spectron Laser Systems

Optional connector (9 pin D-sub connector)

The IO2 card contains an additional connector located on the board with the following pin configuration:

INTERFACE	PIN	ASSIGNMENT
 9 Pin D-Sub male connector	5 9 4 8 3 7 2 6 1	Not connected /OPT4 OPT4 /OPT3 OPT3 /OPT2 OPT2 /OPT1 OPT1

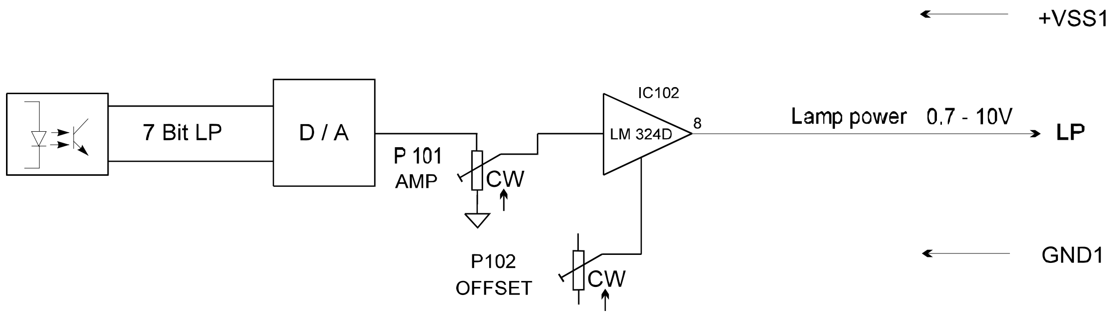
IO-2 Card Technical Notes

Laser Current Control (LP)

The Laser Current Control circuitry delivers a Voltage between 0.7 to 10V. The signal is galvanic isolated from the PC. It can be used to drive power control circuitry of the laser.
The output is controlled through the application software. The WinMCL Plus or PCMarkMT (&Job Editor) programs send to the IO2 via the HC3 card:

- Hex 7F for the lowest output level
- Hex 00 for the highest output level

P102 adjusts the lowest output level:	min 0.7V, max 5V.
P101 adjusts the highest output level:	max 10V.

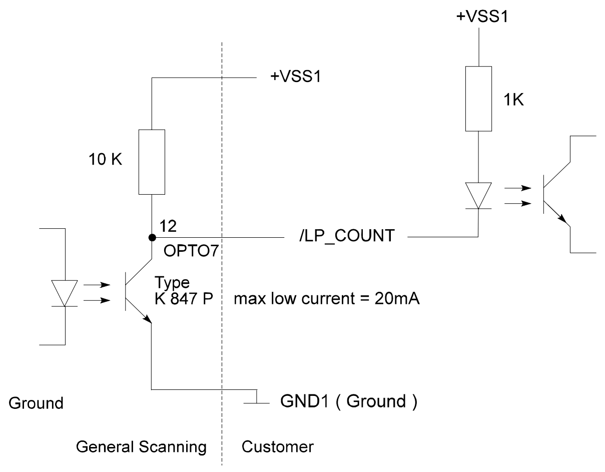


The standard settings are 2V low level, 10V high level. The external power input $^{+}$VSS1 ranges from min $^{+}$15V to $^{+}$18V relative to GND1 with maximum 150mA.

Remote Current Control Flag (/LP_COUT)

The remote current control flag is an opto-isolated output. It indicates that the lamp current circuitry is being controlled through the HC3.

0 = control through the HC3
1 = no control through the HC3



Mode Selection for the Laser HF-Driver

To control the HF-driver through the HC3, it is necessary to select the GATED CW mode on the HF-driver. This is done with the following external mode selection inputs:

- /A = $^+$VCC2
- /B = GND2
-

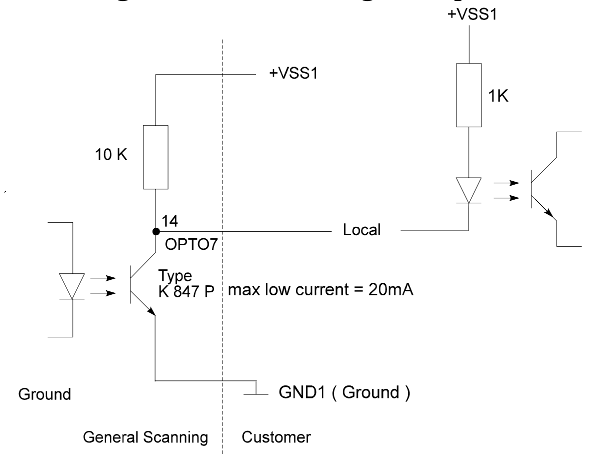
When "REMOTE" is selected on the mode switch (HF-driver) then the external mode selection is activated.

Remote Flag

The remote flag indicates that the mode selection of the HF-driver is used (computer control):

0 = Remote
1 = Local

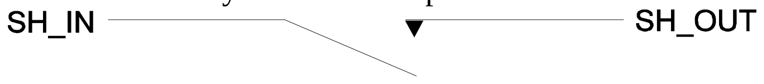
This information is available through the HC3 for application programs (WinMCL Plus, PCMarkMT, etc). At the same time, the Remote flag is available through an opto-isolator as a local flag.



Local = 0V = Remote control through the HC3.

Laser Safety

To improve the laser safety, the Shutter should be connected in series with **SH_IN** and **SH_OUT**. This enables the Shutter control line only when the computer is switched on.



Contact rating:

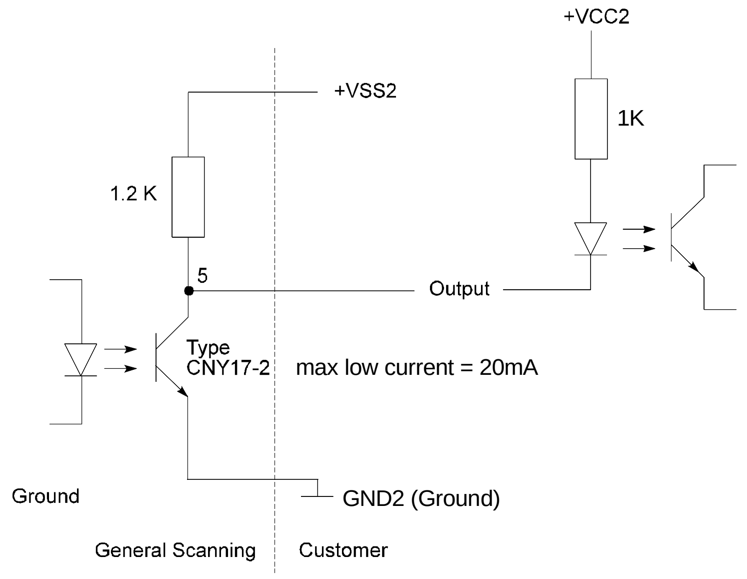
- Max 48V
- Max 0.5A
- Switch on resistance max 200mW

Laser Modulation and First Pulse Suppression

Laser Modulation (LM) and First Pulse Suppression (FPS) outputs have opto-isolators as driver circuitry. The external supply $^+$VSS2 may be between $^+$5V and $^+$18V relative to GND2. The external supply supplies all points labeled $^+$VCC2 and GND2.

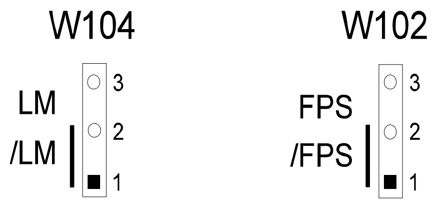
Configuration of the opto-isolator

The /LM-signal is opto isolated through the IC OPTO2 and the /FPS signal through the IC OPTO1.



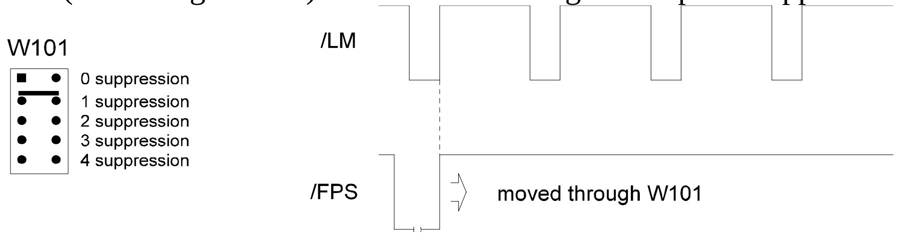
Inverting the Signals

To meet the need of inverting the signals use bridge W102 for the FPS signal and bridge W104 for the LM-signal. Standard settings for LM and FPS are as shown:



FPS Delays

The first pulse suppression can be re configured to a zero, first, and second pulse suppression up to a maximum of 4 pulses (with bridge W101). The default setting is first pulse suppression.

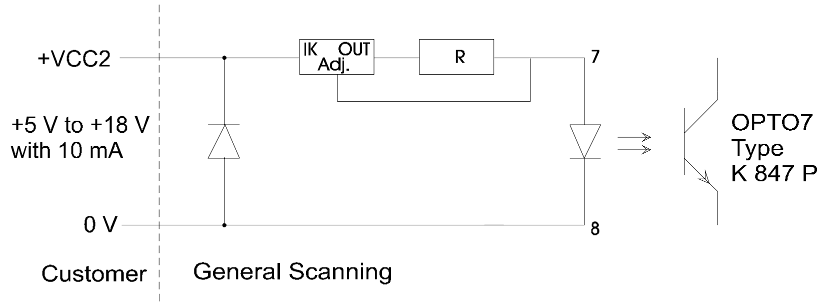


BEGIN_MARK & STOP_MARK

The STOP_MARK and BEGIN_MARK inputs are opto-isolated. The inputs are normally used with an input voltage range from $^+$5V to $^+$18V. The maximum voltage between the input lines should not exceed $^+$18V.

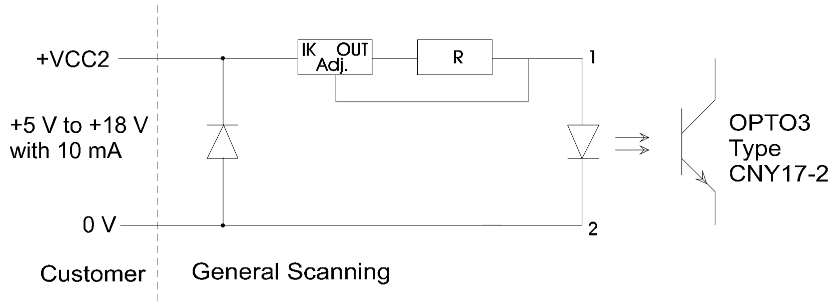
BEGIN_MARK

The BEGIN_MARK input starts the marking through the application programs
(e.g. WinMCL Plus or PC-MARK MT). Configuration of the opto-isolator is as shown:



STOP_MARK

The rising edge of the input voltage triggers the STOP_MARK function. Configuration of the opto-isolator is as follows:



The configuration of the STOP_MARK function can be as follows:

- With the bridge W103 connected between pin 2 & pin 3 used as opto-isolated input for /mark abort, to stop the marking at the end of the stroke.
- With the bridge W103 connected between pin 1 & pin 2 used to stop the Laser Modulation signal immediately and sets /mark abort to stop the marking at the end of the stroke.

By using the bridge connected between pin 1 & pin 2 the STOP_MARK input can be used for trimming applications. The standard setting of bridge W103 is pin 1 to pin 2.

Alarm Flags

All Alarm flags are readable before and after a job is done through the HC3 from the application program. The bits are always low if they are connected and no alarm flag is set. Code definition read back through the HC3:

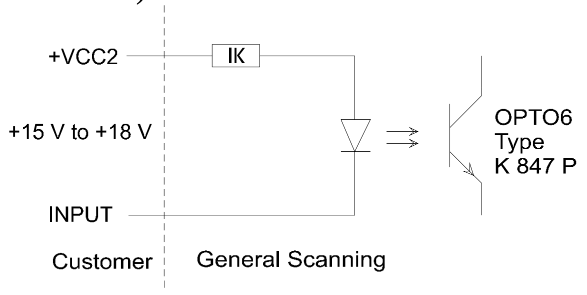
BIT	FLAG
1	/HRP
2	/LFP
3	/MCO
4	/RFO
5	/OPT1
6	/OPT2
7	/OPT3
8	/OPT4

Alarm Flag Input from the HF-Driver

The inputs for the alarm flags are opto-isolated. Sample PIN definition of the alarm flags with Q-switched YAG laser systems[2]:

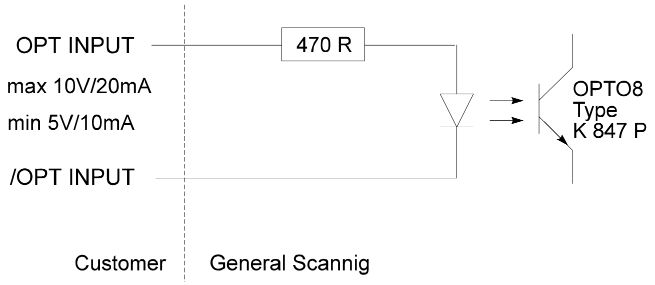
/HRP = high reverse power
/LFP = low forward power
/MCO = modulator crystal over temperature
/RFO = RF modulator driver (heat sink) over temperature

Some Laser Manufacturer's alarms are voltage free normally open contacts. They are closed with no error (closed = healthy, open = on alarm).



Optional Input Flags

The inputs for the optional input flags are opto-isolated inputs. The optional input flags are only available on request of the customer. Configuration of the inputs is as follows:



[2] e.g. Spectron Laser Systems

IO-2 Block Diagram

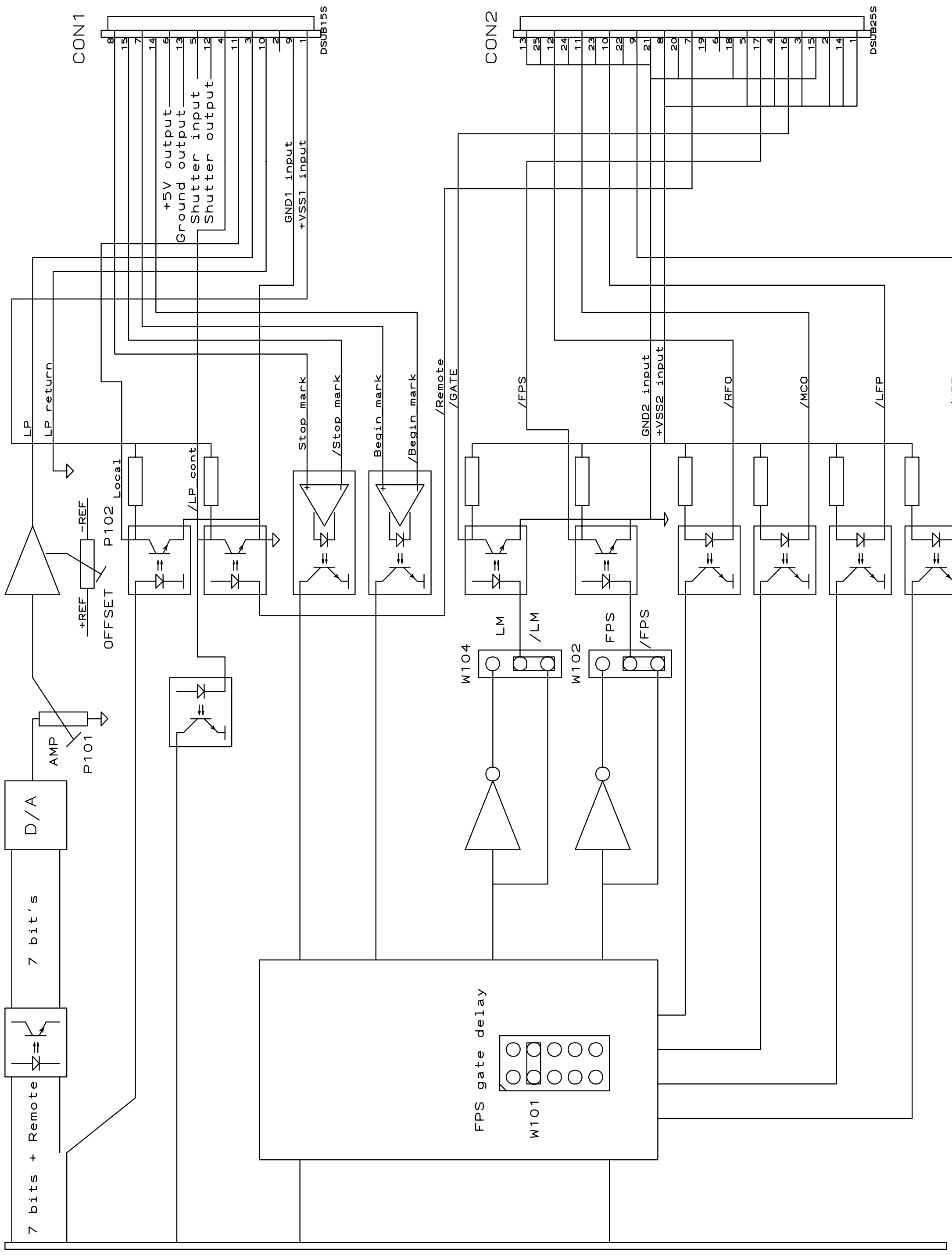


Figure 12: IO2 add-on card block diagram.

END OF DOCUMENT

WinMCL Plus Software User's Manual

References found in this document to prior company and division names such as General Scanning Inc., GSI Lumonics, and GMAX now refer to General Scanning Optical Scanners, a member of GSI Group Inc.

P/N: 7OM-1095
Revision E

www.gs-scanners.com

Contact Information

Americas
39 Manning Road
Billerica, MA 01821
U.S.A.
TEL: +1 (978) 439-5511
FAX: +1 (978) 663-0131
E-mail: ScannerSales-Americas@gsig.com
Toll Free: +1 (800) 342-3757

Europe
Einsteinstrasse 2
D-85716 Unterschleissheim
Germany
TEL: +49 (89) 31707-0
FAX: +49 (89) 31707-250
E-mail: ScannerSales-Europe@gsig.com

Asia
Technoport Kamata, 16-1
Minami-Kamata 2-Chome,
Ohta-Ku Tokyo 144-0035, Japan
TEL: +81 (3) 5425-7733 [Sales]
+81 (3) 5714-0557 [Service]
FAX: +81 (3) 5425-7738 [Sales]
+81 (3) 5714-0566 [Service]
E-mail: ScannerSales-Asia@gsig.com

Table of Contents

1 IMPORTANT INFORMATION

1.1 ESD WARNING

The OEM electronics that *General Scanning* manufactures - including galvanometers and servo controllers - are electrostatic discharge (ESD) sensitive. Improper handling could therefore damage these electronics. *General Scanning* has implemented procedures and precautions for handling these devices and we encourage our customers to do the same. Upon receiving your components, you should note that it is packaged in an ESD-protected container with the appropriate ESD warning labels. The equipment should remain sealed until the user is located at a proper static control station*.

Note: Any equipment returned to the factory must be shipped in anti-static packaging.

(*) A proper static control station **should** include:

1. A soft grounded conductive tabletop or grounded conductive mat on the tabletop.
2. A grounded wrist strap with the appropriate (1 Meg) series resistor connected to the tabletop mat and ground.
3. An adequate earth ground connection such as a water pipe or AC ground.
4. Conductive bags, trays, totes, racks or other containers used for storage.
5. Properly grounded power tools.
6. Personnel handling ESD items should wear ESD protective garments and ground straps.

1.2 Warranty Information

The Customer shall examine each shipment within 10 days of receipt and inform General Scanning of any shortage or damage. If no discrepancies are reported, General Scanning shall assume the shipment was delivered complete and defect free. General Scanning warrantees products against defects up to 1 year from manufacture date, barring unauthorized modifications or misuse. Repaired product is warranted 90 days after the repair is made, or one year after manufacture date - whichever is longer.
Contact Customer Service to obtain a Return Materials Authorization number *before returning any product for repair*.
All orders are subject to the General Scanning Terms and Conditions and Limited Warranty. Visit our website for the latest version of these documents and other useful information.

IMPORTANT: Optical Scanners are normally tuned, serialized and warranted as a matched set for optimized performance. Mismatched components negatively affect performance and void the warranty. A matched set typically consists of galvanometer motor, mirror load, electronic driver board and interface cable.

Customers assume all responsibility for maintaining a laser-safe working environment. OEM customers must assume all responsibility for CDRH (Center for Devices and Radiological Health) certification.

1.3 Customer Support

General Scanning has support services to address your questions or concerns with either the product or manual you are using. Before calling for assistance, be sure to refer to any appropriate sections in the manual that may answer your questions. Call General Scanning's Customer Service Department Monday through Friday between 8 A.M. and 5 P.M. local time (GMT –05:00 Eastern Time (US & Canada)).

The customer service personnel will be able to give you direct assistance and answers to your questions.

INTRODUCTION

General Scanning presents the WinMCL Plus dynamic link library for controlling the HC3 laser and scan control card. WinMCL Plus offers the link between scanner hardware and the user graphical or application interface. WinMCL Plus operates in the Windows NT family including Windows NT, 2000, and XP.

Be careful to observe safety and warnings associated with laser hazards you may encounter. Scan heads consist of galvanometers and attached mirrors that reflect a laser beam whose operation can be hazardous if certain precautions are not taken. Please be alert to the safety considerations and specific procedures regarding the scan head and the laser you are using.

This document features highlighted blue hot-links for rapid navigation through the document. Click on any of the blue text to jump to the associated topic.

SCAN HEAD PROGRAMMING CONSIDERATIONS

Programming laser systems requires special attention to various application issues specific to the laser arena. The following chapter discusses various topics related to using laser systems:

- Marking Jobs
- Fine Tuning with Delays
- Marking Field Transformations
- Field-Distortion Correction
- Laser Power Control
- Mark on the Fly
- Disabling Z Axis
- Hardware I/O
- Separate Signals Mode

Marking Jobs

Marking jobs are accomplished with the three operational modes of the HC/3 card:

- Vector Mode
- Raster Mode
- Shoot Mode

Vector Mode

Vector operational mode is most easily understood by thinking of the beam path as tracing the outline of the object. The beam path is simply the focal point of the mirrors of the marking head on the plane of projection as the GCX file executes in time. During the time that it takes to trace the entire beam path the marking laser may be ON or OFF. When the beam path advances while the laser is OFF, the result appears as if the marker 'jumped' from one location to another; in reality the beam path advanced in a continuous line to the new location but the laser was OFF, consequently there was no marked record of the path of the beam. If the laser is ON while the mirrors are moving then the laser leaves a mark on the object, and these marks are the useful work that the marking head performs. Terminology derived from this example is expressed in the command names, such as MD_jump_rel, a command to move the focal point with the laser OFF, or MD_mark_abs, a command to move the focal point with the laser ON. These are the fundamental operations needed for *vector* mode operations and all other commands and parameters are related to rendering quality or performance tuning.

The WinMCL Plus library provides all the functionality needed to insure that arbitrary projections of the marking head focal point onto the plane of projection can be transformed into a rectangular marking field, where locations within the field of view can be assigned Cartesian coordinates and that this coordinate system can be used in the marking commands. Further discussions in this section will assume that the marking field is rectangular.

In the vector mode of operation a marking job consists of a continuous beam path composed of vectors. The following example illustrates the progression of marking over time by labeling the vertices visited in numerical order.

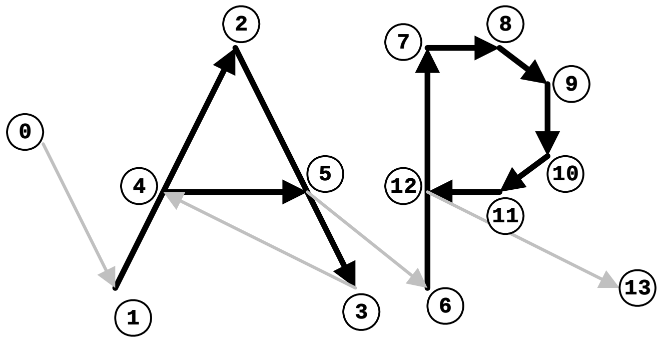


Figure 1: Simple marking path example showing an ordered sequence of jumps and marks.

In the example the arrows point in the direction of beam path motion. The gray arrows indicate jump vectors, motions where the laser is OFF. The black arrows indicate mark vectors, motions where the laser is ON. The example shows the letters 'AP' being marked and the beam path starts at vertex 0 and continues sequentially to vertex 13. The GCX motion commands that implement mark and jump operations are MD_mark_abs and MD_jump_abs when an absolute coordinate system is desired, and MD_mark_rel and MD_jump_rel when a relative coordinate system is desired.

Generally, the coordinate destinations of the mark and jump commands are 16 bit numbers. These are signed numbers that have ranges according to the following table.

# Bits	Min. Value	Max. Value
16	-32768	32767

Position parameters are available for X, Y and Z-axis. The X and Y-axis position values are converted to command voltages that are subsequently sent to power servo controllers operating the mirrors. The Z-axis typically commands a laser focus control, i.e. a linear translator. If you are simply marking on a planar surface then the traditional method of handling the Z parameter is to set each one to zero.

The choice of coordinate system is not an explicit mode of WinMCL, but rather, they are viewpoints to take when constructing a solution to a marking job. There are efficiencies to be had when using relative coordinates, for instance the MCL font characters are defined using the commands MD_mark_rel and MD_jump_rel only. The advantage of this is that the same letter can be marked anywhere in the marking field, without a change to the program. Sequence of instruction is very straightforward, the focal point is jumped (using a jump command and absolute coordinates, i.e. MD_jump_abs) to a location in the marking field that will correspond to the beginning of the letter, then the MCL font marking sequence is run. The operations just described parallel the high level description of the problem; as an example consider the problem:

"Mark the letter 'M' one inch from the top of the pencil."

In order to translate the problem description into a marking sequence we must translate position information into coordinates, and also fill in any missing information or assumptions:

Translate coordinates: "one inch from the top of the pencil" → X = 500, Y = 600, Z = 0

Fill in assumption: "jump the beam to the start point " → MD_jump_abs 500 600 0

Fill in assumption: "Mark the letter 'M'" → There is a marking program for the letter M and it is located in the file C:\MARKING\LETTERS\LET_M.GCX → MD_include "C:\MARKING\LETTERS\LET_M.GCX"

The marking sequence is now fully translated:

```
MD_jump_abs 500 600 0
MD_include "C:\MARKING\LETTERS\LET_M.GCX"
```

The example above assumed that the marking program LET_M.GCX was composed in a relative coordinate system. If this was not an option then the marking operation would be much more complex, as shown in the following example:

```
MD_jump_abs 500 600 0        jump to the start of the character
MD_mark_abs 500 700 0        mark the first riser
MD_mark_abs 550 600 0        mark the first diagonal
MD_mark_abs 600 700 0        mark the second diagonal
MD_mark_abs 600 600 0        mark the first descender
MD_jump_abs 600 600 0        jump to indicate end of mark
```

In order to compare the previous two examples we should expand the file LET_M.GCX into the relative motion commands, so spell out the original marking sequence by expanding all of the MD_include instructions.

```
MD_jump_abs 500  600 0       jump to the start of the character
MD_mark_rel    0  100 0      mark the first riser
MD_mark_rel   50 –100 0       mark the first diagonal
MD_mark_rel   50  100 0      mark the second diagonal
MD_mark_rel    0  -100 0      mark the first descender
MD_jump_rel    0      0 0    jump to indicate end of mark
```

Raster Mode

Raster mode is used for dot rendering of images. There are two MD commands that cause the HC/3 card to render a line of dots, MD_raster_abs and MD_raster_rel. These commands are used in conjunction with jumps and moves to mark a raster screen of dots on the part. Raster mode is different from vector mode in that the timings involved are not based on the Step Period, rather the timing is based on the Q-switch period.

The picture below shows a timeline for 5 pixels in the raster mode. The length of time required to render the raster line is Q-switch period * NumberOfPixels, where NumberOfPixels is a parameter of either the MD_raster_abs or MD_raster_rel commands. In the example, assume that the Q-switch period is 200 µS, thus the length of time required is 5 * 200 = 1000 µS. During the render time the focal point must move from the start position to the end position. Assume that the distance was 500 LSB, thus the beam velocity is 500/1000 = 500000 LSB/sec. It is important to remember to limit the velocity to that appropriate for the servo and galvo.

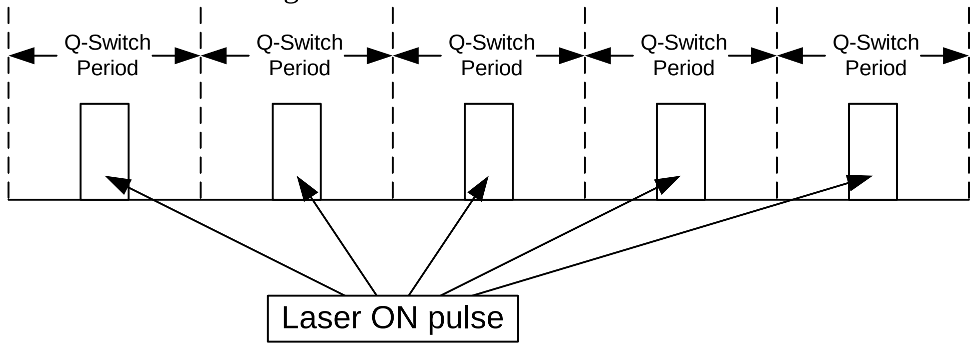


Figure 2: A train of Raster Pulses

The next picture shows the composition of one pixel event in time. The parameters involved are the Laser On Delay, the Laser OFF Delay and the Q-switch period. As previously discussed, the Q-switch period, along with the number of pixels, determine the mark rate. The raster marking commands can generate a 256 level gray scale by modulating the width of the laser control pulse. The laser ON and laser OFF delays determine the baseline intensity of the laser by defining the pulse width of maximum intensity, i.e. the pulse generated by a command of 255. The other laser intensities are proportionally smaller pulse widths in relation to the baseline width. For example, a GSV of 128 generates a pulse width that is half the baseline width.

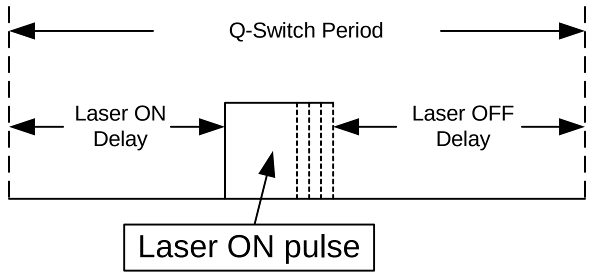


Figure 3: Close-up of timings in a single Raster Pulse

Equivalent definitions of raster pulse width can be defined in any number of ways when using the general form of the definition. The general formula for a GSV command is:

$$RasterPulse_{\max} = (QSwitchPeriod - LaserONDelay - LaserOFFDelay) \times {}^{GSV}\!/_{256}$$

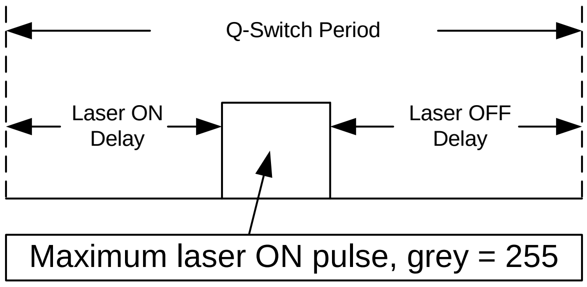


Figure 4: General form of maximum pulse definition

If the Laser On Delay is set to zero then the simplified form of the equation follows:

$$RasterPulse_{\max} = (QSwitchPeriod - LaserOFFDelay) \times {}^{GSV}\!/_{256}$$

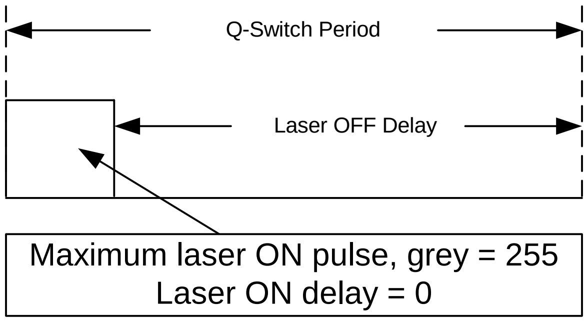


Figure 5: Simple form of maximum pulse definition with Laser ON Delay = 0

Guidelines have been set for consistent operation for the values of the delay parameters:

$$QSwitchPeriod - LaserOffDelay - LaserONDelay > 0$$

So in raster mode the Laser On Delay and Laser OFF Delay operate slightly differently than in vector mode. In vector mode the delays were used to prevent the laser from firing while mirror was moving in its latent time frame with respect to the undelayed MD instruction execution time frame. Raster mode differs in that the delays are used to set the maximum raster pulse that can be commanded in the raster commands. The pacing or raster pulse rate is determined solely by the value of the Q-switch period parameter. The Laser On Delay and Laser OFF Delay simply provide a means to define the width of the maximum raster pulse.

Laser power control for the raster marking instructions can also be output from PortB. There are actually three modes of output of the GSV parameters of the raster commands:

1. Laser power is controlled by the pulse width of the laser modulation signal only.
2. Power control is output via Port B only.
3. Power control is output from both sources.

The diagram shows the timing relationship between of the two outputs. When both outputs are selected, the LM signal can be used simply to trigger a laser pulse and the PortB output controls the laser power with the byte wide digital data.

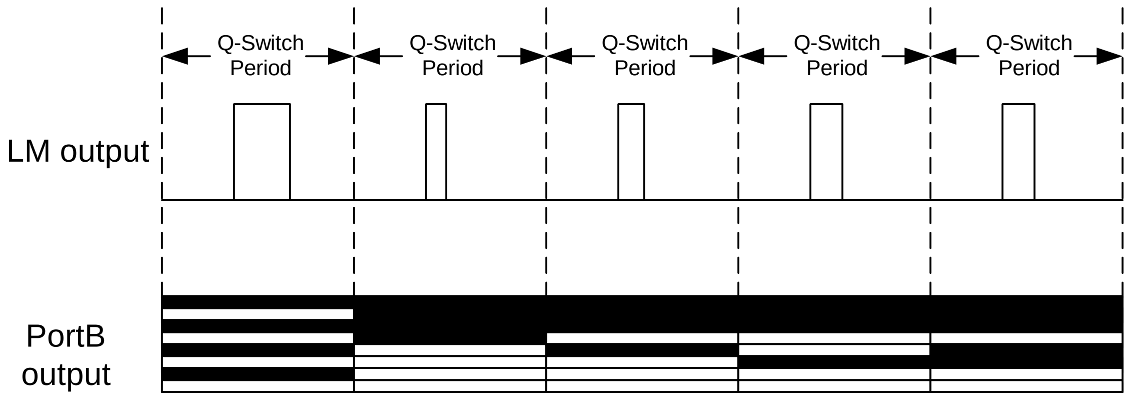


Figure 6: Timing relationship between LM signal and PortB during Raster Pulse generation.

When power only is output from PortB only, the byte value changes at the start of each Q-switch period. Note also that while the raster marking operation is executing, the timing of the command is not based on the StepPeriod, so there is no PowerDelay executed as the PortB data changes. Once the raster marking command completes and vector operation commences, execution of the PowerDelay will once again be enabled and will execute after each change in laser power.
Configuration of the source of laser power control is performed with the MC_set_raster_mode command before a job starts. You must also prepare a parameter set that will invoke the required timings when the raster routine is run:

Q-switch period is configured based on the desired velocity of the focal point during the raster marking routine. Hopefully, the array of raster lines will all be the same length and have the same number of pixels and it is desirable to have the raster beam velocity the same for each line. Compute the velocity as follows, returning the beam velocity in LSB/sec:

$$Beam\,Velocity = \frac{Length\ of\ \ Line}{Number\ of\ \ Pixels\ *\ \ QSwitch\ Period}$$

Laser On Delay and Laser OFF Delay are configured based on the type and power of your marking laser.

The mark rate should be programmed to match the velocity of the raster pulses and a move command issued prior to the raster marking command to establish constant linear velocity. The following diagram shows the 'overscan region' set up by the move command. The parameters required to establish the move mark rate are StepPeriod and MarkSize. Take the value of raster mark rate previously computed and determine a mark rate as the following:

$$Mark\ Rate = \frac{Mark\ Size}{Step\ Period}$$

Since the StepPeriod parameter also affects the jump rate, care should be taken to insure that optimal jump performance is achieved without affecting the matching of the mark rate to the raster marking velocity. Finally, the Stroke Delay should be set to zero, insuring that the transition from the move command to the raster marking command does not have a delay.

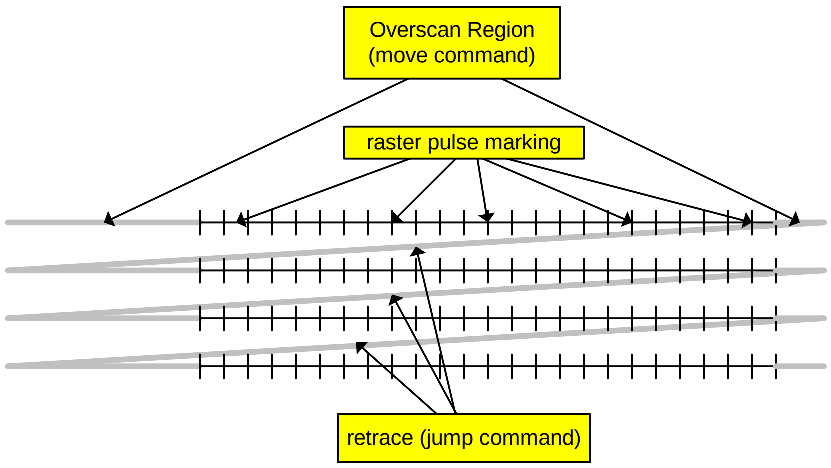
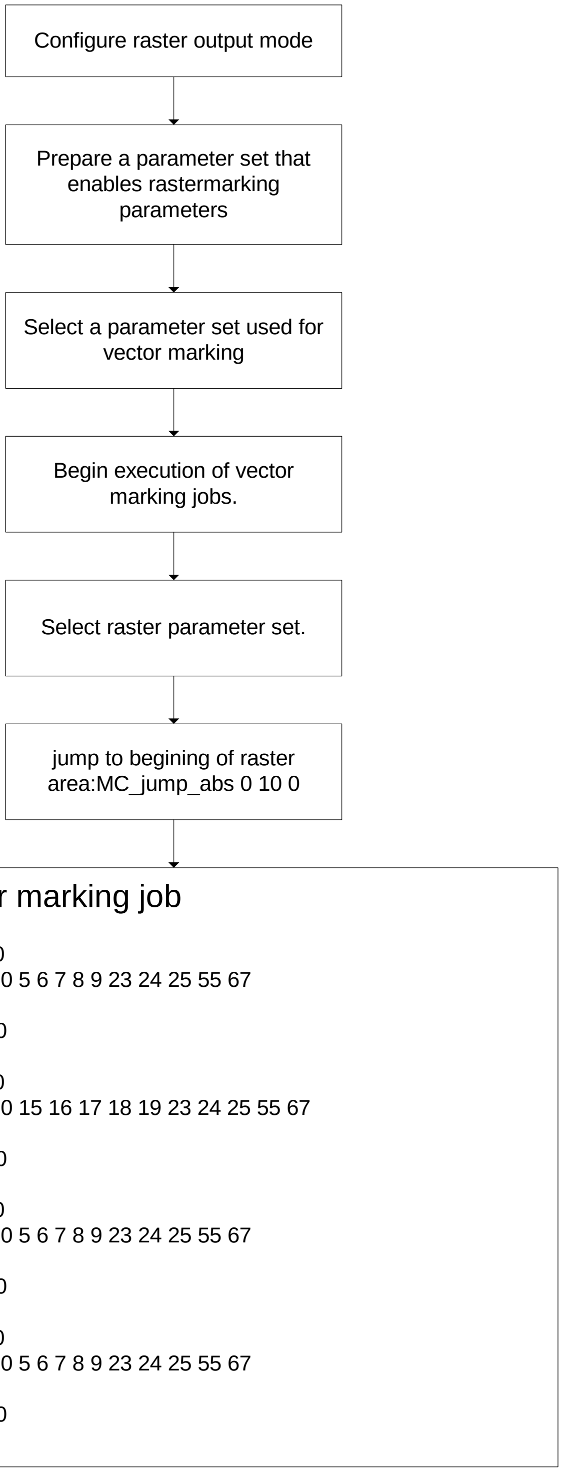


Figure 7: Breakdown of the component motions in a raster marking sequence

The following flowchart outlines the sequence of events needed to mix the above raster marking routine into an overall vector marking routine.


Configure raster output mode
Prepare a parameter set that enables rastermarking parameters
Select a parameter set used for vector marking
Begin execution of vector marking jobs.
Select raster parameter set.
jump to begining of raster area:MC_jump_abs 0 10 0
Execute raster marking job

```
MD_move_rel  10 0 0
MD_raster_rel 20 0 10 5 6 7 8 9 23 24 25 55 67
MD_move_rel  2 0 0
MD_jump_rel -32 -2 0

MD_move_rel  10 0 0
MD_raster_rel 20 0 10 15 16 17 18 19 23 24 25 55 67
MD_move_rel  2 0 0
MD_jump_rel -32 -2 0

MD_move_rel  10 0 0
MD_raster_rel 20 0 10 5 6 7 8 9 23 24 25 55 67
MD_move_rel  2 0 0
MD_jump_rel -32 -2 0

MD_move_rel  10 0 0
MD_raster_rel 20 0 10 5 6 7 8 9 23 24 25 55 67
MD_move_rel  2 0 0
MD_jump_rel -32 -2 0
```

Figure 8: A flowchart that describes the sequence of events when mixing vector and raster marking.

Step and Shoot Mode

Step and shoot mode is most commonly used for the via hole drilling operation of printed circuit production. The operational mode is characterized by a strict sequence of marking commands, jumps followed by a shoot command.

In via hole drilling applications, hole throughput is the prime consideration. The focal point is jumped from target to target; once at a target the jump delay is executed, then the hole is drilled. The special GCX command MD_shoot causes the laser to turn ON for a fixed interval. The following diagram shows the beam path as the job marks:

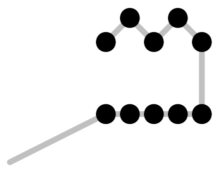


Figure 9: Small portion of a step and shoot marking job

The commands involved are:

- MD_jump_rel
- MD_jump_abs
- MD_shoot

It is important to note that the MD_shoot command must always be preceded by a jump command; this is the defining characteristic of Step and Shoot mode. If the MD_shoot command is executed after any other command, WinMCL will flag an error and stop processing.

The parameters involved in Step and Shoot mode are:

- StepPeriod
- JumpSize
- JumpDelay

StepPeriod and JumpSize determine the Jump Rate, this has to be set as fast as possible without causing undue stability, either in the motion of the mirrors or in mirror shaking when the jump move stops. The JumpDelay is set to allow enough time for the mirrors to stop shaking after the jump.

Of the three operational modes, Vector, Raster and Step and Shoot, Step and Shoot is the only mode that can operate without the use of GCX files. The MC_shoot, MC_jump_abs, and MC_jump_rel functions can be used in a program to implement a via hole drilling program. When operated this way, the jump commands use the timing parameters in the parameter set to establish the jump rate and uses its input parameter <ShootTime> to set the duration of drilling.

Fine Tuning with Delays

An aspect of fine-tuning a marking job is the coordination of the beam path with commanding the laser ON and OFF. Beam positioning lag and laser turn-ON lag are external factors that must be accounted for in software. Various techniques are employed including the addition of delays in the marking instruction stream that can be used to increase the mark rate for a given character quality, or to increase the clarity of the characters. Laser ON and OFF commands are an integral part of the MD_mark and MD_jump commands, there are no specific MD commands for turning the laser ON and OFF and the only means of control is the timing of the hardware. The following introduces the various laser parameters that WinMCL Plus exposes for adjustment by focusing on three key delays listed below, see Chapter 0: Parameter Descriptions for a complete set of adjustable parameters.

- JumpDelay
- Poly Line Delay
- Stroke Delay

The first parameter to be discussed will be the JumpDelay parameter that addresses the lag issues at the end of a jump move. The JumpDelay is part of the Laser Parameter Set and it applies to all jump commands. When the command stream of the GCX file executes a jump instruction the HC/3 card issues a series of voltage outputs that cause the beam position to move at the jump rate. When the final command voltages are sent to the servo the computer assumes that the beam path is located at the new vertex; in fact, the servo may still be steering the mirrors to the new location because of finite servo bandwidth or inertia in the mirrors. WinMCL Plus has a built-in facility to handle this situation, the JumpDelay, and the action is to pause the execution of the GCX file after a jump instruction. The following figure shows that each of the jump vectors has an JumpDelay associated with it.

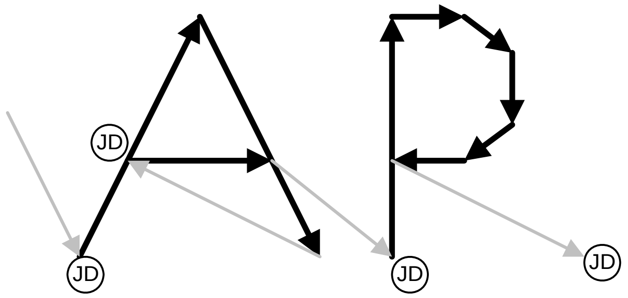


Figure 10: Simple marking path example show the location of Jump Delays

The Laser On Delay timer starts at the beginning of a stroke (the first MD_mark_abs or MD_mark_rel). The Laser On Delay is required to account for finite servo bandwidth, a latency in the mirror time frame with respect to the computer time frame.

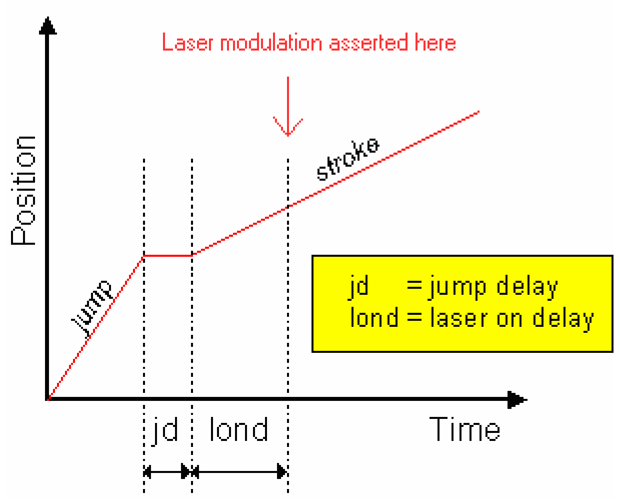


Figure 11: Timing of Jump Delay and Laser ON Delay

The next parameter is called the Poly Line Delay. This delay appears in the beam path after each mark vector of a stroke, except the last. By examining the physics of the mirrors in motion it is possible to understand the utility of the Poly Line Delay. During marking operations the X-axis and Y-axis mirrors move at constant angular velocities that correspond to the beam path that is to be traced out. At each vertex in the stroke the direction of the beam path can change instantaneously, and from the standpoint of commanding the galvos that control the mirrors, the angular velocity must change instantaneously. This is not possible physically due to the inertia of the mirrors, therefore the marked stroke contains an error artifact based on the mechanics of the system. The Poly Line Delay inserts a short pause at each vertex inside the stroke (the last vertex of the stroke is subject to the stroke delay, to be covered next) and the action of this delay is to allow time for the mirrors to attain a new angular velocity. The following figure show the vertices in the example where the Poly Line Delay occurs.

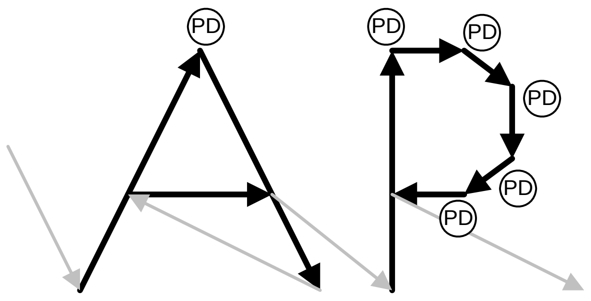


Figure 12: Simple marking path example showing the location of Polyline Delays

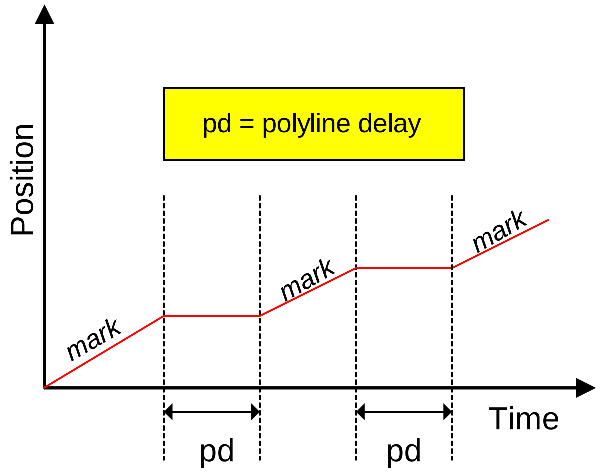


Figure 13: Timing of the Polyline Delay

The final delay parameter used in coordination of the beam path to the laser is the Stroke Delay. The Stroke Delay adds a pause at the last vertex of a stroke. The Stroke Delay is required to account for finite servo bandwidth, latency in the mirror time frame with respect to the computer time frame and the delay allows the mirror to catch up to the commanded position. The eye is more sensitive to shortened strokes than to rounded corners so the stroke delay is included to correct for this.

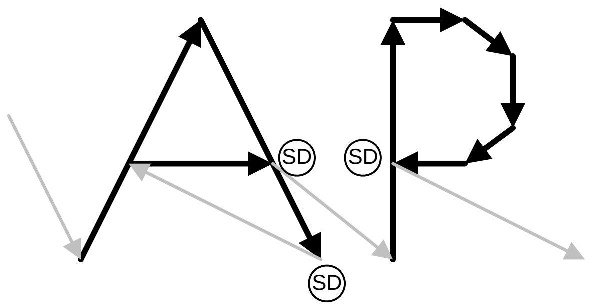


Figure 14: Simple marking path example showing the location of Stroke Delays

The Laser OFF Delay works in conjunction with the stroke delay to control the marking at the end of a stroke, where the laser turns OFF. Due to the real world characteristics of laser gating, some lasers take more time than others to extinguish to an emission level below the marking power. Laser OFF Delay specifies an interval that commences in the stroke delay and ends with the start of the jump move.

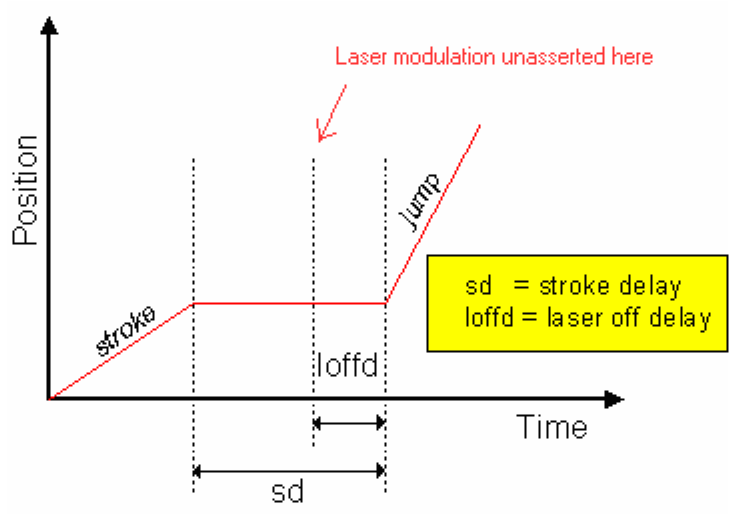


Figure 15: Timing of the Stroke Delay and Laser OFF Delay

Marking Field Transformations

There are five transformation matrixes and Flip/Exchange to be applied to GCX files:

- Object Matrix (OM)
- Part Matrix (PM)
- Global Matrix (GM)
- Universal Matrix (UM)
- Flip Exchange matrix (FM)

First four matrixes are represented as 3x3 matrix of floating point numbers:

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 16: Standard organization for the Transformation Matrices.

The Flip Matrix is represented by the following matrix, where FX and FY are valued either 1 or 0.

$$\begin{bmatrix} 0 & FX & 0 \\ FY & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 17: Organization of the Flip Matrix.

The overall transformation matrix to be applied to the GCX files is calculated as

$$FM * UM * GM * PM * OM$$

There are two types of input methods provided for setting the values of the Global, Object and Part matrices. The 'matrix' input method simply allows you to set the values of certain cells. The names of the parameters in the 'matrix' type MD instruction correspond to the locations in the matrix in the following picture:

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 18: Direct input method form for Transform Matrices.

The second input method is the 'transform' method. This method allows you to input a rotation, in Radians, and an offset, in LSBs. The following picture shows the computation in software that accompanies the 'transform' input method, basically a matrix multiplication of a pure translation matrix with a pure rotation matrix.

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 19: Rotation and translation input form for Transformation Matrices.

As shown above, the '*ang*le' parameter is applied the function, then the appropriate trigonometric function of the value is stored in the matrix. In this manner rotation about the origin of the coordinate system is implemented.

The Object, Part and Global matrices are programmed from within GCX files only. The Universal matrix is programmed from the host software only. The following table shows the commands related to each matrix.

Name	Commands
Object matrix	MD_object_transform MD_object_matrix
Part matrix	MD_part_matrix MD_part_transform
Global matrix	MD_global_matrix MD_global_transform
Universal matrix	MC_set_xform_matrix MC_get_xform_matrix

Field-Distortion Correction

This section tries to clarify the HC/3 operation with respect to grid correction. The HC/3 grid correction is based upon the GMAX N-Point grid correction interpolating algorithm and the GMAX short manual is available as a more complete guide. The NPC.exe program allows fine-tune adjustment of pre-existing grid correction tables that were generated at General Scanning based upon the geometry of GMAX scan head products.

Grid correction is used to compensate for rectilinear errors caused by the nature of the optical beam path, namely the projection of what is essentially a cylindrical or spherical command space onto a flat surface. The primary error is known as pincushion distortion. The following diagram illustrates the maximum extent of the uncorrected X-Y head beam path, showing discrete points and also the theoretical maximum extent in the shape of a hyperbola. Inside of this diagram the red rectangle delineates the maximum the correctable region by adjustment of the commanded coordinates through the feature known as grid correction.

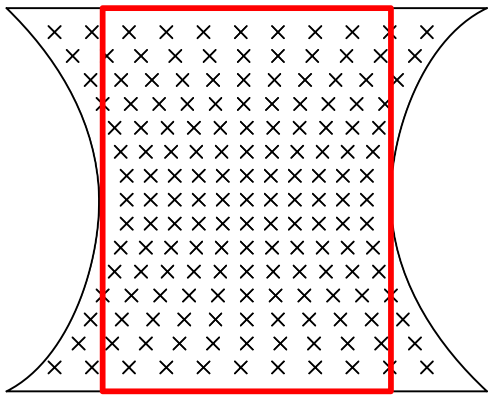


Figure 20: Red area delineates the correctable marking field.

Grid correction simply adjusts the position of points commanded so that they all fall onto a rectilinear grid. Each point has a different X and Y correction factor, but the errors are determined by the physics of the mirror system, so it is possible to correct each point by using a Grid Calibration Table coupled with interpolation.

Grid Calibration Tables are provided with each GMAX head for a given focal length and lens. These tables are loaded into the WinMCL Plus system using the commands MC_set_corrtable and MC_set_corrtableFromMemory.

Laser Power Control

The HC/3 card provides two hardware interfaces for programmatic control of laser power, PortB and the Laser Modulation signal. The value output by PortB is controlled by the Power parameter. Each time the value changes output occurs on both Port B of the isolated connector and Port B of the internal connector. The Power Delay is executed on each change, to compensate for the time frame latency between the laser and computer. The Power parameter is actually an index into a WinMCL data structure called the Power Table.

Operations like MC_init and MC_reset cause the default power table to be loaded and the value of the Power parameter to be set to zero, then the zero value is output to Port B, the Power Delay executed if the port was outputting other than zero before. The value sent to Port B can be changed through several different operations.

If the value of the power parameter changes, it will index a new cell in the power table. If the value in the cell is different than the value at Port B then the cell value, as pointed to by the new power parameter value, will be sent to Port B, and the Power Delay then executed because Port B change in value.
If the Power parameter remains constant but the power table changes then the cell in the power table referenced by the power parameter may contain a different value, if so, the value of the cell is sent to Port B and Power Delay is executed.

In vector mode the laser power commanded during a stroke is always constant. Use the Power parameter in the Parameter Set to set the laser power to a specific value during the ON period.

In multi-card systems it is possible to calibrate the power of multiple lasers so that the resultant marks are the same for different lasers. Use the power table to calibrate the various lasers to the commanded value. The power table format consists of 256 entries, where the value indexed by the Power parameter is the value sent to the Port B output, the laser power control output.

There are several commands that manipulate the Laser Power Table. These are:

- MC_get_powertable
- MC_reset_powertable
- MC_set_powertable

MC_get_powertable allows you to retrieve a power table for a given HC/3 card. This can be useful if you want to make minor adjustments to a certain power setting without affecting other power table settings.

MC_reset_powertable is a function that resets the power table to the default values (1:1 correspondence), as shown in the table above.

MC_set_powertable is used to assign a previously allocated power table data structure to an HC/3 card.

Related Topics:

- MC_get_powertable
- MC_reset_powertable
- MC_set_powertable

- Port B output
- Laser Power Calibration Table
- Laser Power Calibration Table default value

Mark on the Fly

The Mark on the Fly feature allows the scan head to follow and mark an object moving with constant velocity through the scanning field, such as items on a conveyor-belt. Before the mark job for an object is started, the application informs WinMCL of the object's velocity in the X and Y directions. Using this information WinMCL changes the output stream opcode to follow the object movement.

The following MCL function calls are related to Mark on the Fly:

- MC_get_target_velocity
- MC_set_target_velocity
- MC_reset_tracking
- MC_start_tracking

The following MD commands are related to Mark on the Fly:

- MD_start_tracking
- MD_reset_tracking

Disabling Z Axis

When running two axis systems, the Z-axis should be disabled. While the scan head will simply ignore Z-axis signals, one can get up to one-third increase in maximum throughput with this axis disabled as significantly less information must be transmitted for each vector.

A registry entry controls the Disable Z-axis function. By default, the Z-axis is disabled. The following procedure allows you to adjust the Disable Z-axis status:

1. Open regedit.exe
2. Navigate to HKEY_LOCAL_MACHINE/SOFTWARE/GSILumonics/WinMCL32
3. Modify the "DisableZ" variable to 1 to disable, 0 to enable.

Hardware I/O

The HC/3 card has two I/O connectors that break out user signals. I2 is a 9-pin connector located on the computer bulkhead. This connector provides optically isolated inputs and outputs.

	Name	**Pin Number**
I2 signals	Opto-isolator GND	1
	Laser Modulation	2
	MARK_ABORT signal	3
	BEGIN_MARK signal	4
	MARK_IN_PROGRESS	5
	Opto-isolator +5 VDC	6
	OPT2 signal	7
	REMOTE_EXECUTE signal	8
	MARK_ERROR signal	9

The second connector provided is the internal connector I3. The signals on this connector are not optically isolated. In addition to the signals provided on I2, there are several other signals and ports available on I3.

	Name	**Pin Number**
I3 signals	Port A input	1-8
	Port B output	8-16
	BEGIN_MARK signal	17
		18
		19
	MARK_ABORT signal	20
	MARK_IN_PROGRESS	21
	Shutter 1 in	22
	FPS	23
	Shutter 1 out	24
	Ground	25
	Ground	26
	+5V	27
	+5V	28
	-12V	29
	+12V	30
	Clr	31
	step	32
	Laser Modulation	33
	4 MHz	34
	Port N output	35-50

Separate Signals Mode

Historically the laser on/off and modulation would be controlled by the same signal. The laser would be off and would not emit power if no modulation was present. With the laser on, the Laser Modulation signal would oscillate according to the Q-switch period and Q-switch frequency. Hence the Laser Modulation signal acts as both laser on/off gate as well as Q-switch modulation (also called pulsed modulation) source.

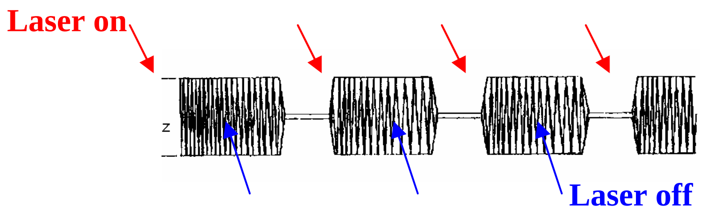


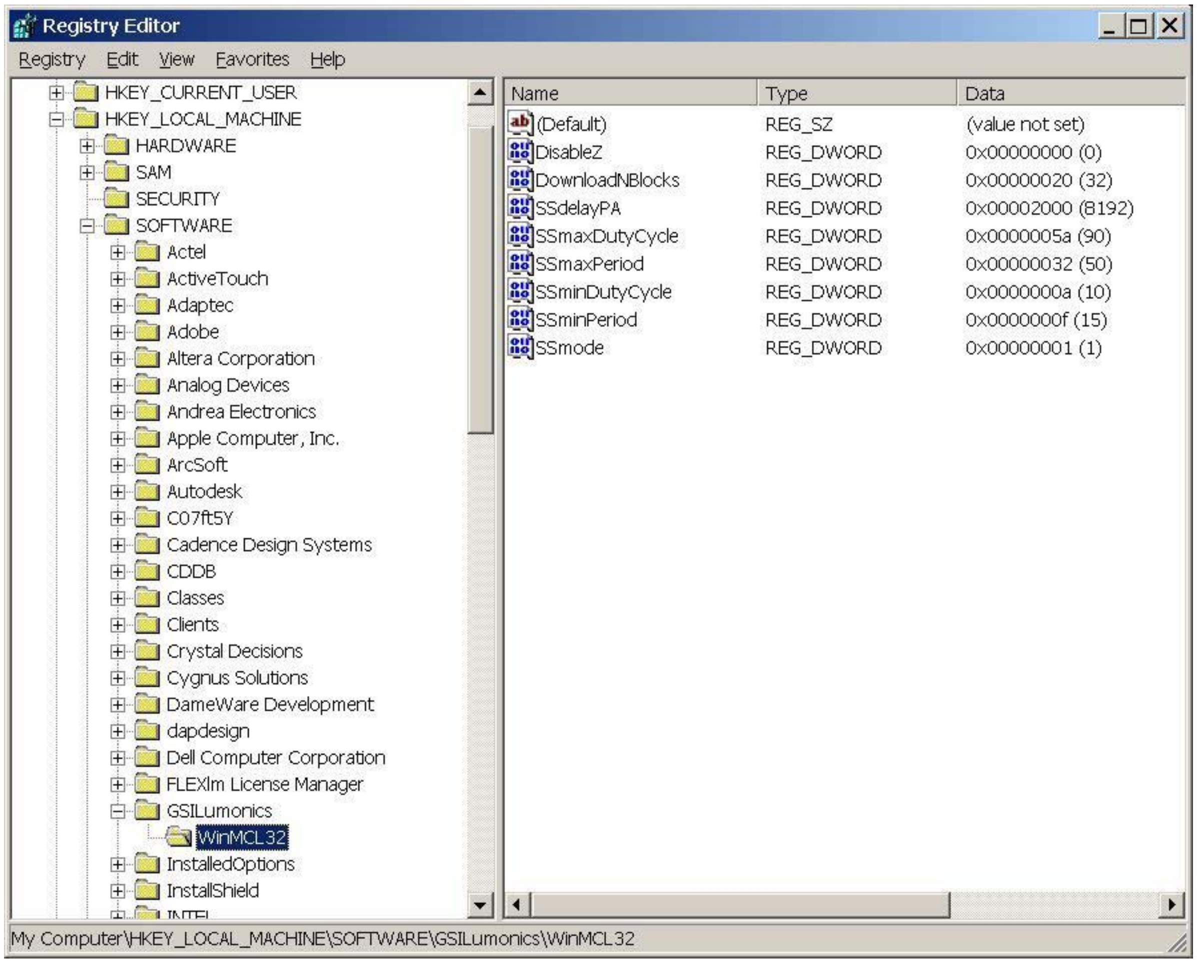
Figure 21: Traditional Laser Modulation Signal

In the separate signals mode the gate mode and Q-switch modulation are removed from each other. The simply high/low laser on/off gate is controlled through the FPS pin while the Q-Switch frequency and period remain on the LM pin.

Note that inverting the FPS signal (see Section 0 GCX Test Program and Figure 31: HC/3 configuration dialog box) will invert whatever the output is on the FPS pin (the pin labeled as FPS). While in Separated Signal Mode if the FPS signal is inverted then the gate signal is also inverted.

To enable the separate signals mode the registry must be changed. Change SSmode equal to 1 to activate the separate signals mode, 0 disables separate signal mode. The default SSmode is 0 or off. The registry also includes settings for constraining the output signals while in this mode. SSmaxDutyCycle, SSmaxPeriod, SSminDutyCycle, and SSminPeriod limit the rate of oscillation of the output LM signal. See the Q-switch width and Q-switch period parameters for the definitions of the laser modulation oscillation output.

Many lasers have a delay between the time the laser starts to oscillate and the time the laser outputs power from the cavity. In most lasers the power is applied to the cavity at the same time the laser oscillator is activated, as seen in Figure 21. Some lasers carry warnings not to apply power to the cavity until the oscillator has sufficient time to build up energy. In those cases where the laser should be modulated before the gate signal is applied, the SSdelay registry sets a delay between the start of the LM signal and the latch of the gate (FPS pin) signal.



Name	Type	Data
(Default)	REG_SZ	(value not set)
DisableZ	REG_DWORD	0x00000000 (0)
DownloadNBlocks	REG_DWORD	0x00000020 (32)
SSdelayPA	REG_DWORD	0x00002000 (8192)
SSmaxDutyCycle	REG_DWORD	0x0000005a (90)
SSmaxPeriod	REG_DWORD	0x00000032 (50)
SSminDutyCycle	REG_DWORD	0x0000000a (10)
SSminPeriod	REG_DWORD	0x0000000f (15)
SSmode	REG_DWORD	0x00000001 (1)

My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\GSILumonics\WinMCL32

Figure 22: Separate Signals Mode registry settings.

PROGRAMMING WINMCL PLUS

The following section serves as a tutorial for those engaging in the programming of WinMCL Plus.

DLLs and LIBs on the WinMCL Plus CD-ROM

The WinMCL Plus library is stored in WinMCL32.DLL. Several other DLL's are distributed to support utilities such as NPC.exe. These other DLL's need not be present to use WinMCL Plus functions in your own programs.

Microsoft compilers expect library files in the Common Object File Format, and will link with WinMCL32lib. Borland compilers and others that use the Object Module Format can link with WinMCL32.omf.lib.

Support Programs GcxAsm, NPC, Postgrid, wgcxview, and WinMCLtest32

The WinMCL Plus includes a variety of support programs for use in scanning applications including GcxAsm, NPC, Postgrid, wgcxview, and WinMCLtest32. None of these programs need be present in integrator developed front-end applications.

GcxAsm allows users to open a list of MD/MC commands in text format and assemble them into a gcx file. The GcxAsm will check for errors and report the findings. The gxc file may then be loaded in WinMCLtest32 to be executed. See section 0 GCX File Assembler Test Program for more information.

NPC, or N-Point Calibration, enables users to do grid correction to account for geometric and other errors associated with scanning systems. The NPC yields a good quality correction after two or three iterations. The NPC process results in an .asc file that should be loaded into the grid table field of WinMCLtest32 (be sure to press the set button). Several correction files for GSILumonics scan heads and industry standard lenses are included in the cal folder of the WinMCL Plus installation, so using NPC may not be necessary. See the NPC manual in the doc folder for more information.

Postgrid creates the z-axis coordinates and grid correction file for General Scanning three-axis HPLK systems. Postgrid will output an .asc file to be loaded into WinMCLtest32 or the front-end application. The Postgrid program will come on a separate disk with the HPLK system, although the latest configuration files (.con) are located in the cal directory of WinMCL Plus. For more information about Postgrid, see the manual included with the Postgrid installation.

Wgcxview enables viewing of .gcx files, an invaluable tool when evaluating recently created gcx files. Simply open the desired .gcx file to view the gcx output. Jump movements, where the laser is not on, may be toggled by pressing the jump hot button on the toolbar. The jump toolbar button appears as an arc with an arrow on it.

WinMCLtest32 provides a straightforward interface for executing gcx files and otherwise testing the control of a HC3 system. WinMCLtest32 is not designed to aid the user in creating a front-end, only for debugging one. For more information on WinMCLtest32, see section 0 GCX Test Program.

General Comments on Function Calls

The WinMCL Plus API consists of functions that are used to interface the user program to the HC3 card. In general, the function interface is similar for each of the functions and they differ in the quantity and type of parameters and, of course, in the operations that they perform.

Function parameter lists are typically organized with the input parameters preceding the output parameters. The MC_get_version function is an exception.

Input parameters are usually passed by value unless the parameter is a string or a structure. For example, the CardIndex parameter is used in many functions to select one HC/3 card in an installation that contains multiple cards. The CardIndex is passed by value. A file name is a parameter passed to several functions. The file name is stored in a string and a pointer to the string is passed to the function.

Output parameters appear in the function parameter list giving the WinMCL Plus functions the ability to return multiple values. These are always pointers to variables, located either in the calling function or in the global scope.

The return value of a function in the WinMCL Plus API is always the error status of the operation, either WMCL_OPERATION_OK or WMCL_OPERATION_FAILED (as defined in WinMCL32.h). The exceptions are as follows:

- Functions that return a value other than an error code: MC_get_timeout.

- Functions that return only WMCL_OPERATION_OK: MC_get_error.

The return value of the functions should be read back by the calling code and, in the event of an error, the calling procedure should check the WinMCL Plus global error code by calling the function MC_get_error.

A variety of data types are used as parameters, but real numbers are always stored in the single precision size (float).

GCX Files

GCX files and GCX formatted buffers contain the motion commands that describe a marking job. The format of GCX op-codes in a text file mirrors the format of the binary GCX file, but in terms of WinMCL functions that execute GCX files, a GCX file is binary only; any text versions of GCX have to be submitted to an assembler for conversion to binary GCX format. The following example will use the assembly language of GCX file for the examples, but it is assumed that the files will be converted into binary before being submitted to WinMCL for execution.

The simplest GCX files consist of a block or list of GCX motion commands. The following example shows several types of operations that are performed with GCX files.

A simple marking routine

```
MD_list_begin                       ; This delimiter is always needed
MD_jump_abs   0       0       0     ; Jump to the origin of the marking field
MD_jump_abs   9000    0       0     ; Jump to the beginning of the graphic feature

MD_mark_rel   -2250   -5000   0     ; Begin to mark the feature
MD_mark_rel   -4500   10000   0
MD_mark_rel   -2250   -5000   0
MD_mark_rel   -3000   0       0
MD_mark_rel   50      50      0
MD_mark_rel   50      0       0
MD_mark_rel   50      -50     0
MD_mark_rel   300     60      0
MD_jump_abs   0       0       0     ; Jump to indicate end of marking
MD_list_end
```

The simple marking routine illustrates several important features about all GCX files and buffers. The list of instructions is delineated by the MD_list_begin and MD_list_end commands. All other instructions must appear inside these delimiters. The delimiters show the extent of one 'job'. Jobs are loaded sequentially onto the job queue, execution of the job queue is suspended pending the invocation of the MC_start_mark or MC_start_mark_on_begin functions. Once the queue is started the jobs are executed in the order in which they were pushed onto the job queue.

A common technique when marking on a planar marking field is to set the Z-axis = 0 for all motion commands.

Notice that the job does not contain any instructions related to delays like the stroke delay, polyline delay, etc. These delays are generated as a side effect of the instructions in the stream and they do not and cannot be explicitly inserted into a marking job but they can be programmed from the parameter set.

A Step and Shoot Marking Job

This is an example of a Step and Shoot program. It is important to remember that the MD_shoot command must always be preceded by a jump command; otherwise WinMCL will generate an error when processing the file. The file shows some common operations such as selecting a parameter set before drilling.

```
MD_list_begin
MD_select_parameter_set 4
MD_jump_abs    0     0     0

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    500   500   0

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_jump_rel    10    0     0
MD_shoot       25

MD_list_end
```

A marking routine that uses Mark on the Fly

```
MD_list_begin
MD_select_parameter_set 0           ; Select a parameter set that was prepared using
                                    ; WinMCL commands

MD_jump_abs     0       0           0

MD_mark_rel     0       5000        0
MD_mark_rel     0       -10000      0
MD_mark_rel     0       5000        0

MD_stop_tracking                    ; Stop the tracking counters

MD_reset_tracking                   ; Reset the tracking counters

MD_list_end
```

A routine that renders a 4-line raster graphic

```
MD_list_begin
MD_select_parameter_set     34      ; Set parameters
MD_jump_abs     50      50      0   ; Move to the start of the feature

MD_move_rel     100     0       0   ; Execute a lead-in over scan region
MD_raster_rel   50      0       10  255  255  255  255  255  0 0 0 0 0 ; (Note 1)
MD_move_rel     25      0       0   ; Execute a lead-out over scan region
MD_jump_rel     175     -2      0   ; Execute a retrace

MD_move_rel     100     0       0
MD_raster_rel   50      0       10  255  255  255  255  255  0 0 0 0 0
MD_move_rel     25      0       0
MD_jump_rel     175     -2      0

MD_move_rel     100     0       0
MD_raster_rel   50      0       10  255  255  255  255  255  0 0 0 0 0
MD_move_rel     25      0       0
MD_jump_rel     175     -2      0

MD_move_rel     100     0       0
MD_raster_rel   50      0       10  255  255  255  255  255  0 0 0 0 0
MD_move_rel     25      0       0
MD_jump_rel     175     -2      0

MD_list_end
```

(Note 1); Mark the raster data

GCX File Syntax

<GCX file>	**{<GCX list> | <include block> | <comment block> | <include data record> | <transform matrix record>}+**
<include data record>	MD_include_data<pointer> <uint>
<include block>	<include begin record> <include text> <include end record>
<include begin record>	MD_include
<include end record>	MD_include_end
<include text>	<file name>
<file name>	{<file name character> <file name character>}+
<file name character>	(any UNICODE character valid for file names in Win-dows NT}
<GCX list>	<list begin record> [<comment block>][<list block>]<list end record>
<list begin record>	MD_list_begin
<list end record>	MD_list_end
<list block>	[<comment block>] [<para set selector record>][{<jump record>}+ | {<move record>}+ | {<mark record>}+ | {<shoot record>}+
<transform matrix record>	{MD_global_matrix | MD_part_matrix | MD_object_matrix <matrix coefficients>} | {MD_global_transform | MD_part_transform | MD_object_transform <theta_x_y coefficients >}
<comment block>	<comment begin record> <comment text> <comment end record>
<comment begin record>	MD_comment_begin
<comment end record>	MD_comment_end
<comment text>	{<comment character> <comment character>}*
<comment character>	(any non-NUL UNICODE character)
<para set selector record>	MD_select_parameter_set <parameter set index>
<parameter set index>	0 ... 63
<jump record>	MD_jump_abs | MD_jump_rel <vector data>
<move record>	MD_move_abs | MD_move_rel <vector data>
<mark record>	MD_mark_abs | MD_mark_rel <vector data>
<shoot record>	MD_shoot <int value>
<vector data>	<int value> <int value> <int value>
<matrix coefficients>	<float value> <float value> <float value> <float value> <float value> <float value>
<theta_x_y coefficients>	<float value> <float value> <float value>
<bool value>	0,1
<ushort value>	0 ... 65535
<int value>	-2147483648 ... +2147483647
<unit value>	0 ・4294967295
<float value>	
<pointer>	(A valid 32bit address in the application's memory space.)

Data Structure Definitions

Error Code Structure

Defined in winmcl32_api.h.

```
#define ERROR_MSG_LENGTH 255
typedef struct _ERROR_STRUCT
{
    unsigned long Code;
    char Msg[ERROR_MSG_LENGTH];

} ERROR_STRUCT;
```

Note: This structure is used with the MC_get_error function.

Parameter Set Structure

```
struct LASERPARAMETERSET
{
        unsigned long BreakAngle;
        unsigned long FPSEnable;
        long TargetXVelocity;
        long TargetYVelocity;
        unsigned long Power;
        unsigned long PowerDelay;
        unsigned long QSwitchPeriod;
        unsigned long QSwitchWidth;
        unsigned long Intensity;
        unsigned long StepPeriod;
        unsigned long JumpSize;
        unsigned long JumpDelay;
        unsigned long MarkSize;
        unsigned long PolyLineDelay;
        unsigned long StrokeDelay;
        long LaserOnDelay;
        unsigned long LaserOffDelay;
        unsigned long DitherWidth;
        unsigned long DitherFeed;
};

typedef LASERPARAMETERSET PSET;
```

Defined in winmcl32_api.h

Note: This structure is used with the following functions and commands:
MC_get_ParamSet
MC_reset_ParamSet
MC_select_parameter_set
MC_set_ParamSet
MD_select_parameter_set

Grid Correction Table

```
#define GRID_SIDE   65                  // One side of the correction grid.
#define GRID_SIZE   GRID_SIDE * GRID_SIDE   // Grid is 65 by 65 values.

typedef struct _CORRTABLE
{
    long XData[GRID_SIZE];
    long YData[GRID_SIZE];
    long ZData[GRID_SIZE];

} CORRTABLE;
```

Defined in winmcl32_api.h

Power Table

Defined in winmcl32_api.h

```
#define POWER_POINTS 256

typedef struct _POWERTABLE
{
    unsigned long
PowerCorrect[POWER_POINTS];
} POWERTABLE;
```

Parameter Sets

The WinMCL Plus software is capable of storing 64 parameter sets in memory. The parameter set structure is defined on the next page.

The following list of WinMCL commands operate on the parameter sets:

- MC_set_ParamSet
- MC_get_ParamSet
- MC_reset_ParamSet
- MC_select_parameter_set

```
struct LASERPARAMETERSET
{
        unsigned long BreakAngle;
        unsigned long FPSEnable;
        long TargetXVelocity;
        long TargetYVelocity;
        unsigned long Power;
        unsigned long PowerDelay;
        unsigned long QSwitchPeriod;
        unsigned long QSwitchWidth;
        unsigned long Intensity;
        unsigned long StepPeriod;
        unsigned long JumpSize;
        unsigned long JumpDelay;
        unsigned long MarkSize;
        unsigned long PolyLineDelay;
        unsigned long StrokeDelay;
        long LaserOnDelay;
        unsigned long LaserOffDelay;
        unsigned long DitherWidth;
        unsigned long DitherFeed;
};

typedef LASERPARAMETERSET PSET;
```

A procedure to operate on parameter sets might be to:

1. Allocate space on the heap for a parameter structure.
2. Store the parameter set in the parameter set array by calling the MC_set_ParamSet function.
3. Copy in values from the default value parameter set using the MC_reset_ParamSet function.
4. Adjust the values of parameters of most concern by operating with the original pointer from allocation.
5. Make the value available for use by selecting the parameter set with the MC_select_parameter_set function.

See also:
- List of all Parameters
- Parameter Set default values

WinMCL Error Codes

Name	Number	Description
Responses from API functions		
WMCL_OPERATION_OK	0	General response, if no error.
WMCL_OPERATION_FAILED	1	General error response.
WMCL_BUSY	2	WMCL is busy.
General error codes		
WMCL_INVALID_COMMAND	100	Invalid command entered by API function.
WMCL_TIMEOUT	102	No response from thread function.
WMCL_NOT_INITIALISED	103	MC_init() has not been called yet.
WMCL_CREATE_EVENT_FAILED	111	Unable to create Win32 event.
WMCL_REGISTER_EVENT_FAILED	112	Unable to register event with driver
Installation / HC/3 errors		
WMCL_NO_HC3_CARD_AT_ADDRESS	200	Magic code not read.
WMCL_ONLY_ONE_INSTANCE	201	Only one instance of WMCL allowed.
WMCL_HC_ACCESS_ERROR	202	HC/3 access error.
WMCL_NO_HCDRV	203	Could not open HCDRV.
WMCL file / memory errors		
WMCL_GCX_FILE_NOT_FOUND	300	MCL vector file not found.
WMCL_DATA_OVERFLOW	301	If data is too high or too low.
WMCL_WRONG_GCX_FILE_FORMAT	302	Error while reading WMCL file.
WMCL_MEMORY_ERROR	303	Not enough memory for FIFO.
WMCL_FILE_MAPPING_ERROR	304	Memory mapped file error.
WMCL_HC_QUEUE_ERROR	309	If download to kernel mode queue failed
WMCL_CORR_FILE_NOT_FOUND	311	Correction table file not found.
WMCL_WRONG_CORR_FILE_FORMAT	312	Correction table file format wrong (LT).
WMCL_LASER_PARAMETER_BOUNDS	313	A parameter value was outside the legal bounds.
WMCL_POWER_FILE_NOT_FOUND	315	Power correction table file not found.
WMCL_WRONG_POWER_FILE_FORMAT	316	Power table file format wrong (PT).
WMCL_DPL_POWER_OUT_OF_RANGE	318	
WMCL_WRONG_INCLUDE_DATA_FORMAT	319	INCLUDE_DATA block is missing begin or end magic value.
WMCL_BAD_INCLUDE_DATA	320	INCLUDE_DATA block address cannot be read.
WMCL_BAD_DATA_POINTER	321	GCX data block cannot be read.
WMCL_EVENT_RESET_FAILED	322	Unable to reset win32 event.

Name	Number	Description
WMCL file / memory errors		
WMCL_SYSTEM_ERROR	323	System error.
WMCL_OUTPUT_QUEUE_ERROR	504	HC/3 Queue error.
WMCL_OUTPUT_FIFO_ERROR	505	HC/3 FIFO error.
WMCL_OUTPUT_FIFOSTART_ERROR	506	HC/3 FIFO start error.
WMCL_OUTPUT_SOFTWARE_ABORT_ERROR	507	Software abort error.
WMCL_OUTPUT_UPCALL_ERROR	508	HC/3 FIFO upcall error, now deprecated.
WMCL_OUTPUT_UNEXPECTED_F2_ERROR	509	HC/3 FIFO unexpected F2.
WMCL_OUTPUT_UPCALL_FIFO_EMPTY	510	
WMCL_OUTPUT_UPCALL_FIFO_FULL	511	
WMCL_OUTPUT_UPCALL_FIFO_FUNC1	512	
WMCL_OUTPUT_UPCALL_FIFO_HALTED	513	
WMCL_OUTPUT_UPCALL_FIFO_MARK_ABORT	514	HC hardware abort error.
WMCL_OUTPUT_UPCALL_UNEXPECTED_INT_REQ _STATUS	515	

Obsolete MC and MD Commands

Many of the depreciated MC and MD commands from earlier versions of WinMCL are now obsolete. The following is a list of these commands.

Obsolete MC Commands

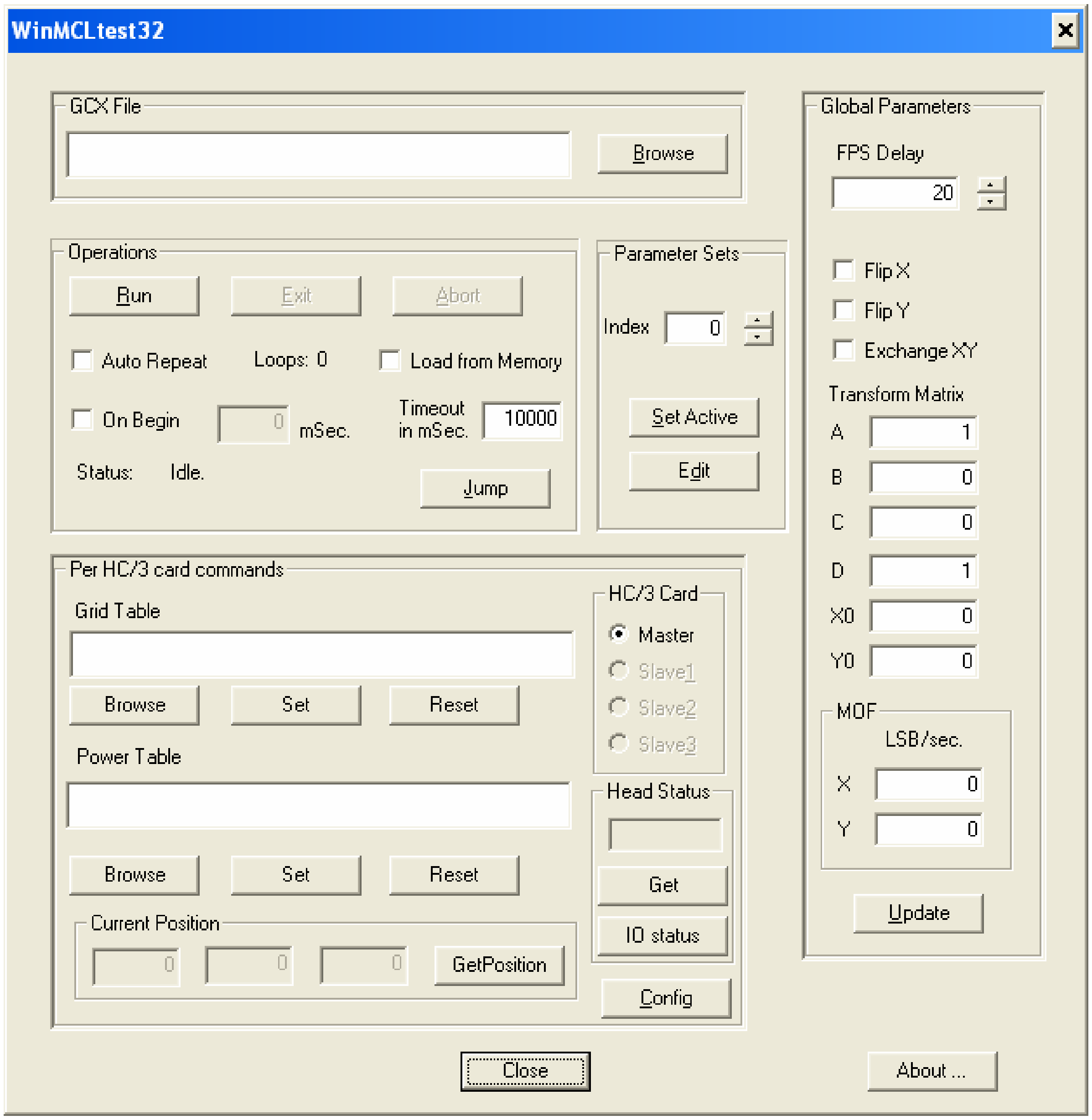
- MC_do_mark
- MC_do_mark_blocking
- MC_end_mark
- MC_get_Corrtablename
- MC_get_Progress_count
- MC_get_corrfile
- MC_get_cur_intensity
- MC_get_cur_power
- MC_get_cur_x
- MC_get_cur_y
- MC_get_cur_z
- MC_get_do_mark_status
- MC_get_fps_period
- MC_get_load_job_status
- MC_get_power_limits
- MC_get_power_scale
- MC_get_power_scale_ factor
- MC_get_powertablename
- MC_get_start_mark_status
- MC_load_job_blocking
- MC_load_job_from_ memory_blocking
- MC_register_Progress_ callback
- MC_reset_Progress_count
- MC_run
- MC_set_cur_z
- MC_set_fps_period
- MC_set_power_limits
- MC_set_power_scale
- MC_set_power_scale_ factor
- MC_start_mark_on_begin_blocking
- MC_stop

Obsolete MD Commands

- MD_line_abs
- MD_line_rel
- MD_line_rle
- MD_progress_marker
- MD_set_autosegmentation
- MD_set_beam_dump_position
- MD_set_shoot_time
- MD_set_wait_for_laser_off
- MD_set_wait_for_laser_on

GCX Test Program

WinMCL comes with a simple test program that can be used to load and execute GCX files and it is useful for testing the hardware and software installation. The application makes use of WinMCL Plus and it is an example of the ease with which functional applications can be built when using WinMCL Plus.



Loops: 0
Timeout in mSec. 10000
Status: Idle.
Index 0
HC/3 Card: (● Master)
FPS Delay 20

A	1
B	0
C	0
D	1
X0	0
Y0	0

MOF LSB/sec.: X 0, Y 0

Figure 23: The main window of the WinMCLTest32 program.

The program allows selection of GCX, Grid Correction and Power Table files, execution of GCX files gated by the BeginMark signal, direct input of parameter set values and global parameter values and geometry transform values.
Typing in the file path in the box specifies GCX files. Alternately, you can press the Browse button to raise a file dialog box.

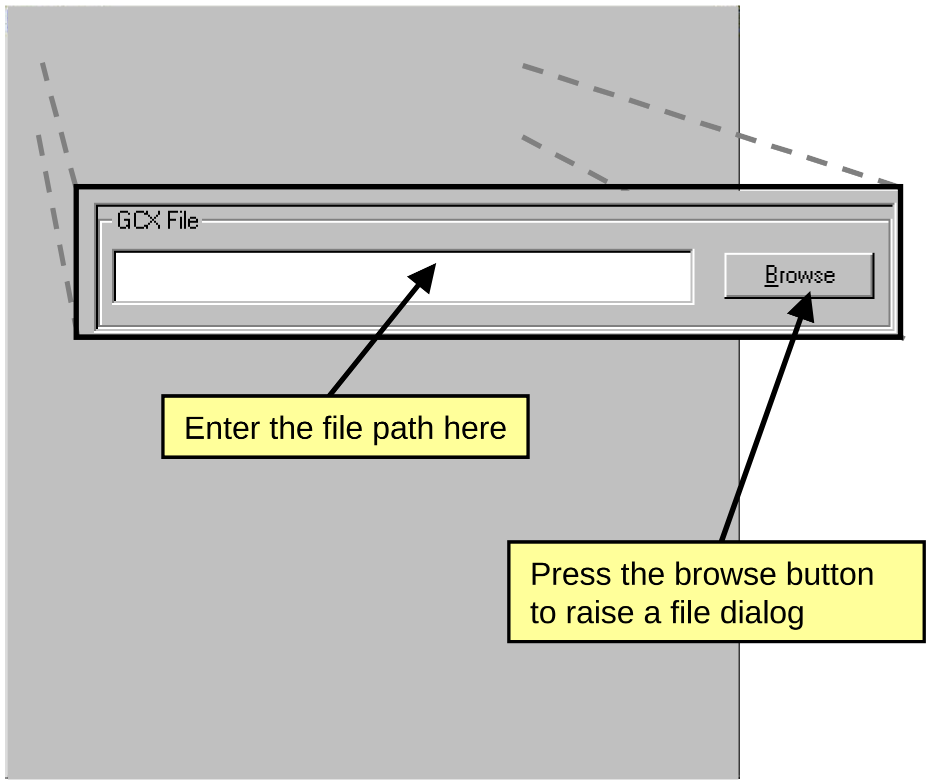


Figure 24: How to specify the GCX file to run.

Before a GCX file is run it is important to configure the system to operate properly. Please read through this section completely to understand what settings have to be made before a job is run.

The job contained in the GCX file is executed when the Run button is pressed. The program has controls that allow configuration of the job execution. The Auto Repeat checkbox is used to configure repeating behavior for GCX files. First the Auto Repeat box is checked, then the Run button is pressed. The job contained in the selected GCX file is run and at the end it is reloaded and run once again. This repeats endlessly until pressing the Exit or Abort button terminates it. The Exit button simply prevents the job from being reloaded; when pressed, the job runs to completion and then terminates normally. The Abort button causes the MC_abort_mark command to terminate the job immediately. It is important to note that the MC_abort_mark function will leave the position and laser control signals in an undefined state.

The program provides a way to trigger a marking job via the BEGIN_MARK signal. Select the GCX file to run and then check the On Begin checkbox. When you press the Run button the job will not start, instead the program will 'arm' waiting for the BEGIN_MARK signal to become asserted. Once BEGIN_MARK signal asserts the job will commence.

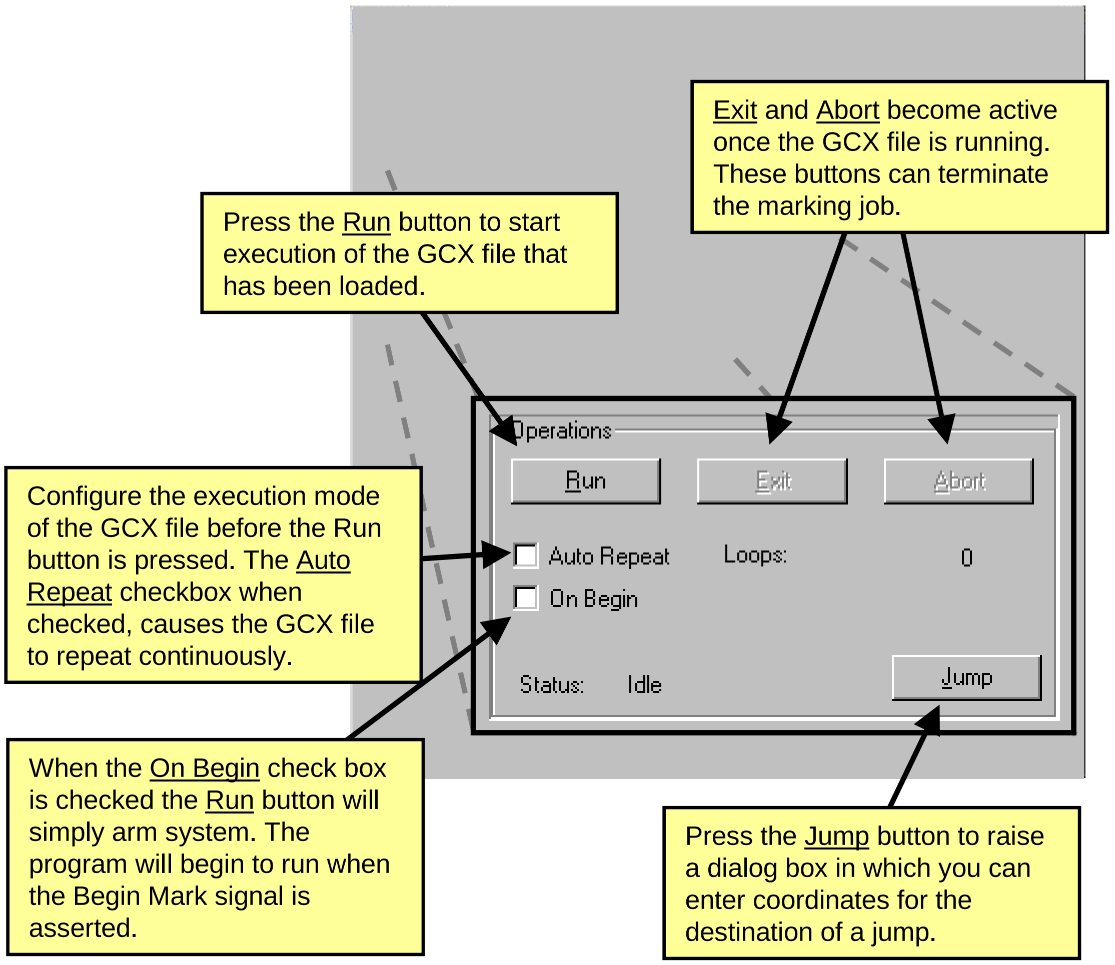

Exit and Abort become active once the GCX file is running. These buttons can terminate the marking job.
Press the Run button to start execution of the GCX file that has been loaded.
Operations
Run
Exit
Abort
Configure the execution mode of the GCX file before the Run button is pressed. The Auto Repeat checkbox when checked, causes the GCX file to repeat continuously.
Auto Repeat
Loops:
0
On Begin
Status:
Idle
Jump
When the On Begin check box is checked the Run button will simply arm system. The program will begin to run when the Begin Mark signal is asserted.
Press the Jump button to raise a dialog box in which you can enter coordinates for the destination of a jump.

Figure 25: GCX file execution options.

The program also provides a mechanism to move the focal point with a jump command. When the Jump button is pressed the jump dialog box appears. This contains input fields for the X, Y and Z coordinates of the jump destination and a configuration checkbox that allows you to declare the coordinates to be absolute or relative.

Once the coordinates have been entered press OK to execute the jump command. The focal point will move at the jump rate as defined by the StepPeriod and JumpSize parameters of the selected parameter set. If you decide not to jump, simply press the Cancel button to return to the main window.

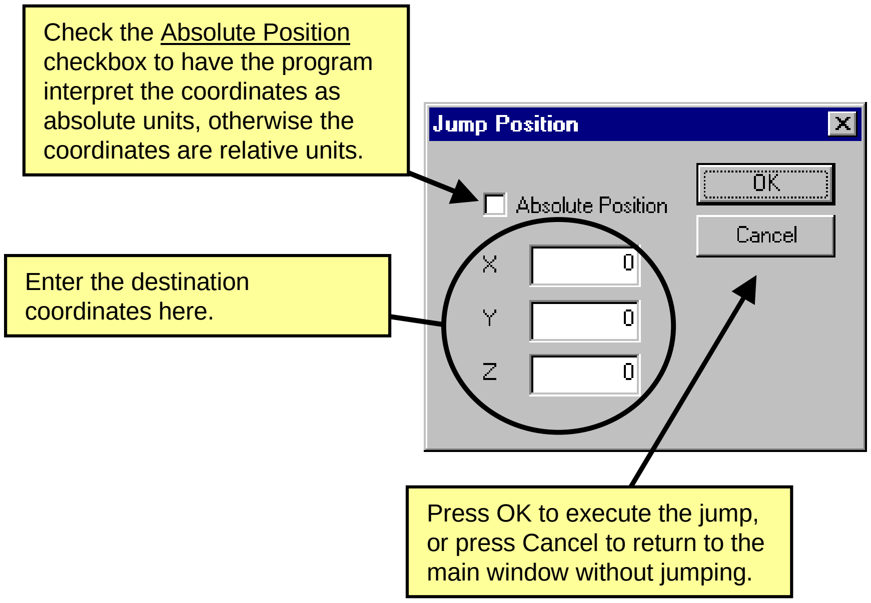

Check the Absolute Position checkbox to have the program interpret the coordinates as absolute units, otherwise the coordinates are relative units.
Jump Position
Absolute Position
OK
Cancel
X
Y
Z
0
0
0
Enter the destination coordinates here.
Press OK to execute the jump, or press Cancel to return to the main window without jumping.

Figure 26: The Jump dialog box.

Parameter sets can be selected and edited. To select a parameter set use the Index control to set the index of the desired parameter set, then press the Set Active button. To edit any parameter set use the Index control to point to the desired one, then press the Edit button. The parameter dialog will appear and display the current parameter values in the set. When the program is first started, parameter set 0 will be selected by default.

Important note: The Index control does not always indicate the index of the selected (active) parameter set. Please remember which set you selected, especially if you are editing a number of different parameter sets.

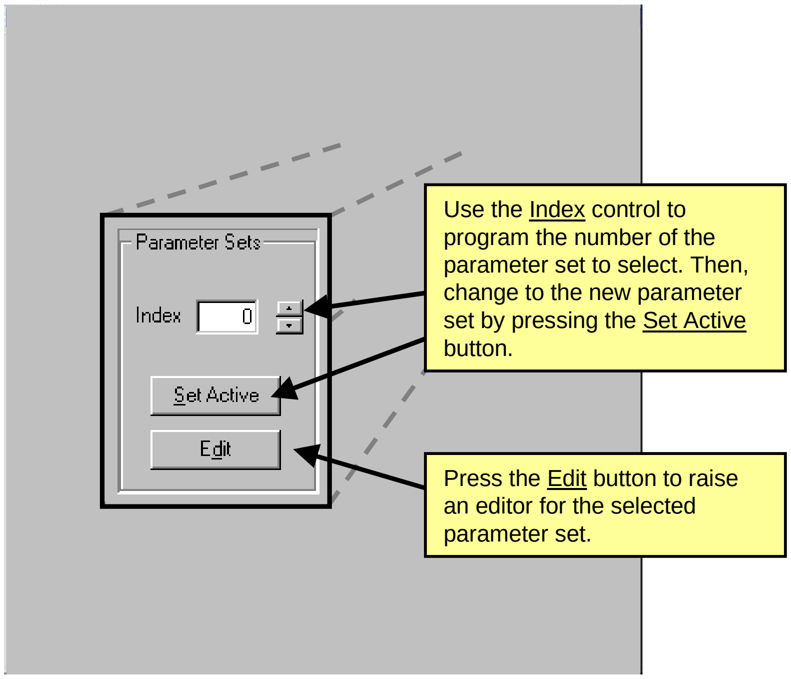
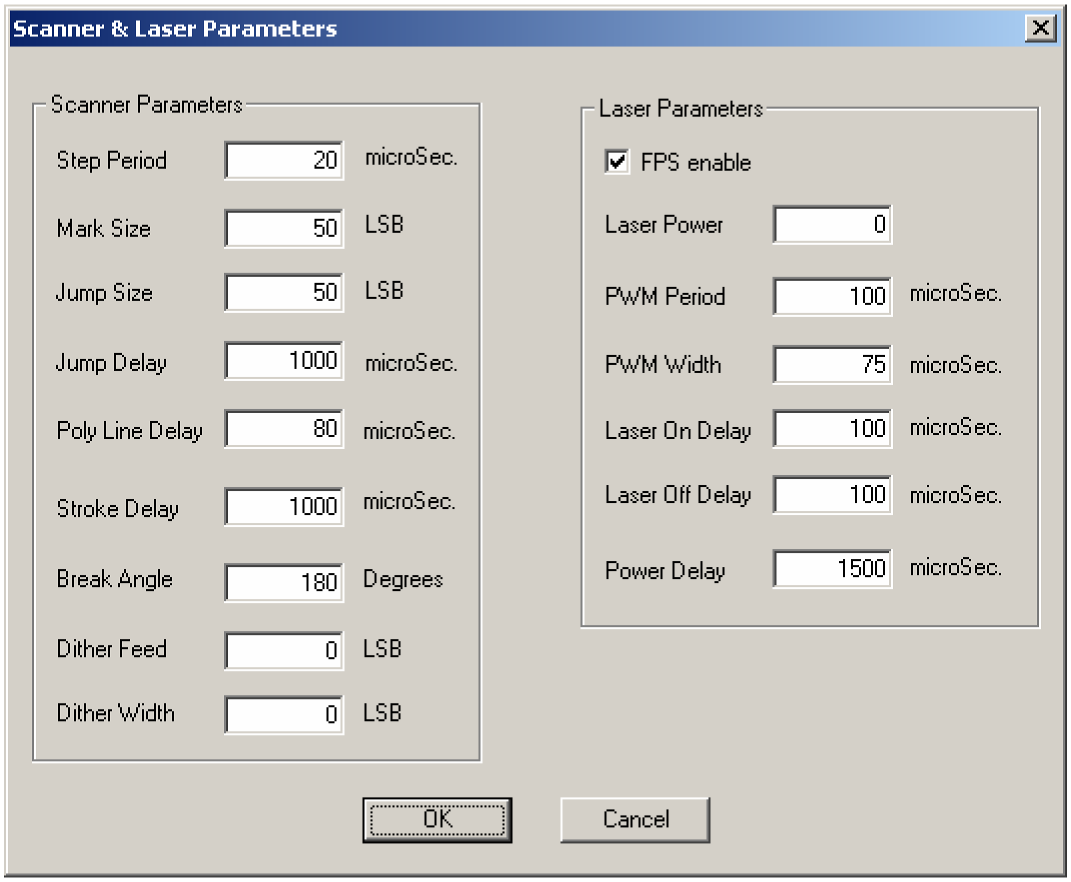

Parameter Sets
Index
0
Set Active
Edit
Use the Index control to program the number of the parameter set to select. Then, change to the new parameter set by pressing the Set Active button.
Press the Edit button to raise an editor for the selected parameter set.

Figure 27: Controls that work with parameter sets.

The parameter editor has input fields for each of the parameters. If you enter a value out-of-range for a given parameter an error message will pop up when you press the OK button. When OK is pressed, the parameter set is saved in the program. To select it as the selected parameter set press the Set Active button.

When the program is first started all 64 parameter sets are loaded with the same default values.



Scanner Parameters

Step Period	20	microSec.
Mark Size	50	LSB
Jump Size	50	LSB
Jump Delay	1000	microSec.
Poly Line Delay	80	microSec.
Stroke Delay	1000	microSec.
Break Angle	180	Degrees
Dither Feed	0	LSB
Dither Width	0	LSB

Laser Parameters

[✓] FPS enable

Laser Power	0	
PWM Period	100	microSec.
PWM Width	75	microSec.
Laser On Delay	100	microSec.
Laser Off Delay	100	microSec.
Power Delay	1500	microSec.

Figure 28: The parameter set dialog box

The Global Parameters are programmed with input fields located on the main window. Make the changes as needed and then press the Update button to register the values with the program. When the program is first started, the values of the Global Parameters are given suitable default values.

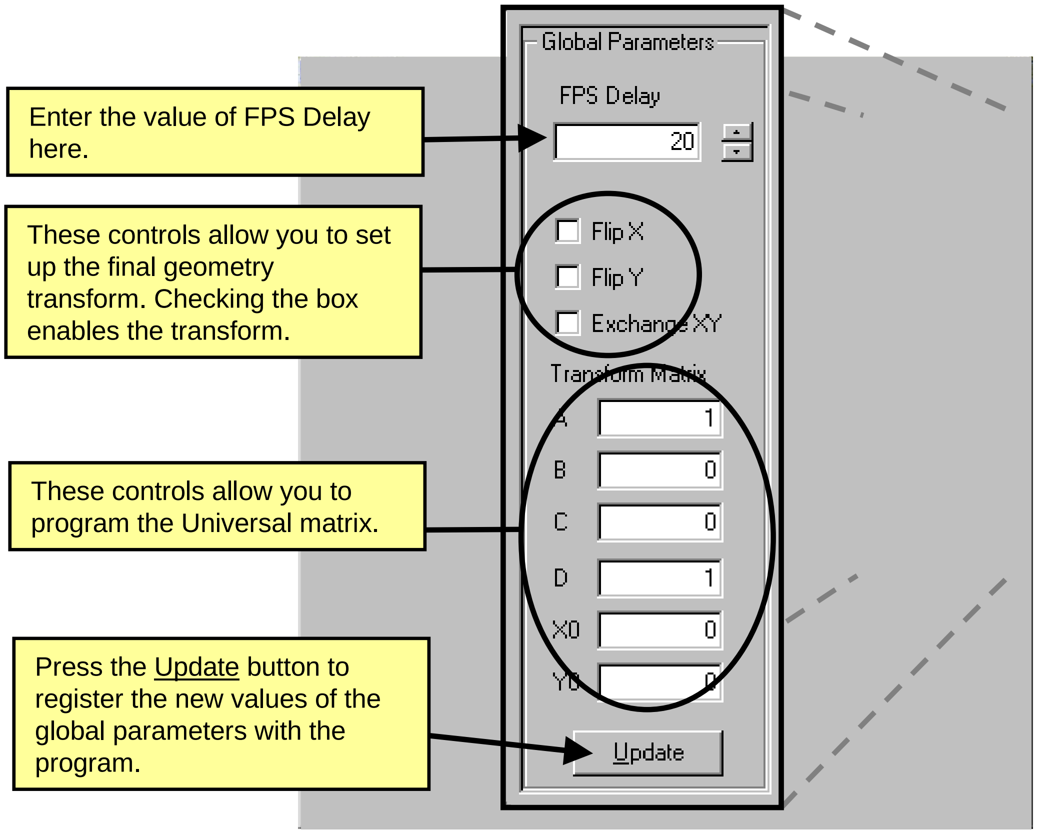

Enter the value of FPS Delay here.
These controls allow you to set up the final geometry transform. Checking the box enables the transform.
These controls allow you to program the Universal matrix.
Press the Update button to register the new values of the global parameters with the program.
Global Parameters
FPS Delay
20
Flip X
Flip Y
Exchange XY
Transform Matrix
A 1
B 0
C 0
D 1
X0 0
Y0 0
Update

Figure 29: Controls that work with Global Parameters

Some installations have more than one HC/3 card. An aspect of these systems is that each card has it's own Grid Correction Table and Power Table. The program shows you how many cards are recognized by enabling a certain number of radio buttons. These radio buttons are used to select the destination for configuration changes. To program the two tables on a per card basis you must first select the HC/3 card with the radio button and then enter a file name or browse for both the Grid Table and Power Table. Once data is complete, press the Set button in the Grid Table section and in the Power Table section.

When the program is first started the Grid and Power Tables take on the default values for each installed card, i.e. tables that do not provide any correction.

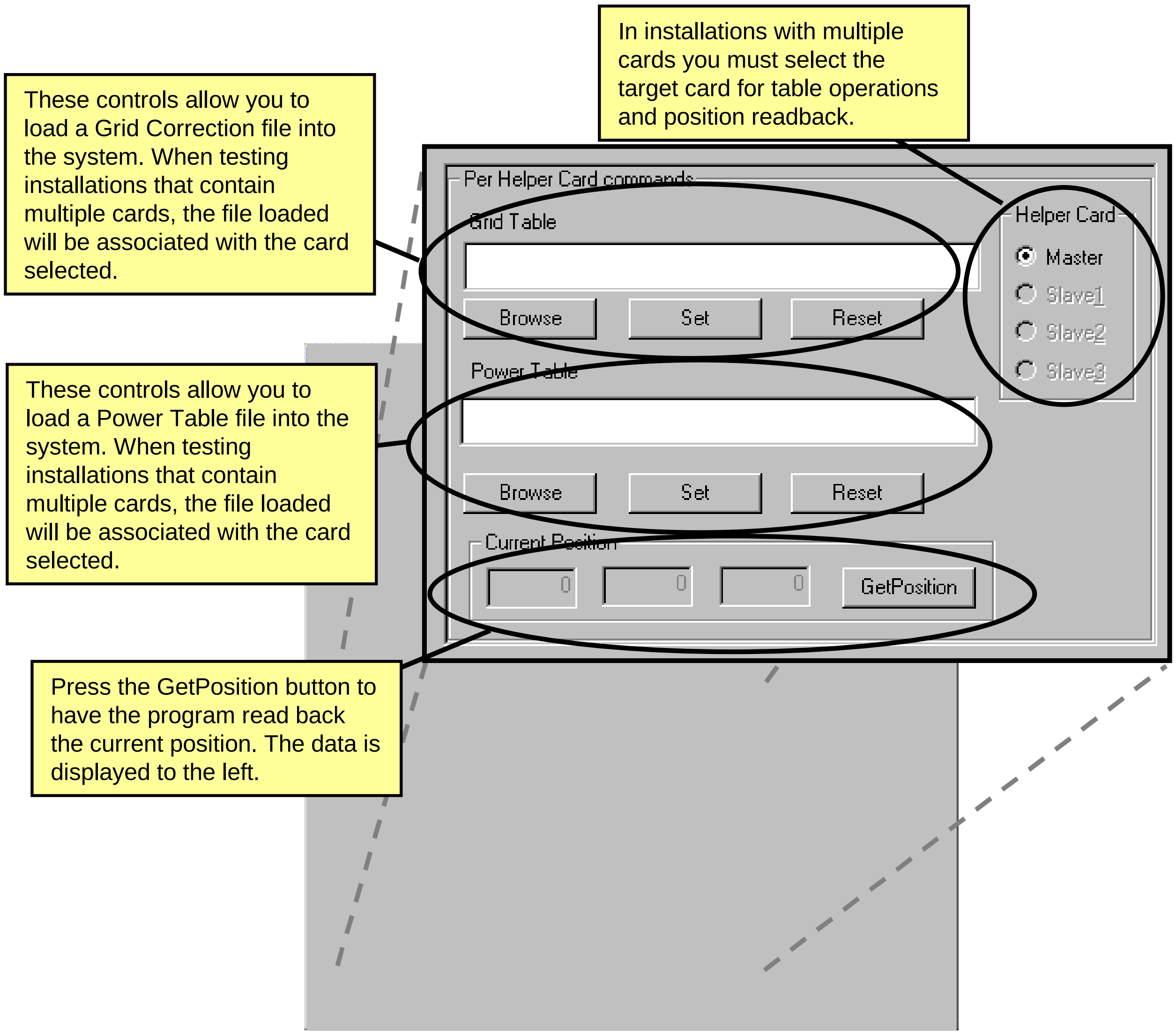
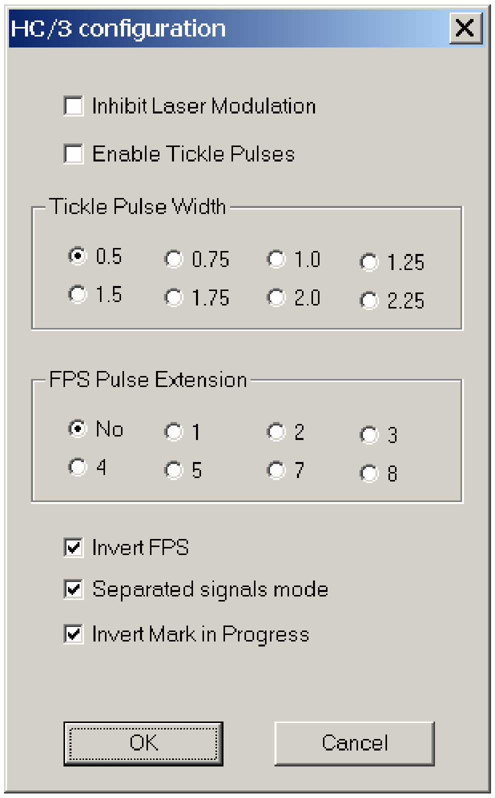

In installations with multiple cards you must select the target card for table operations and position readback.
These controls allow you to load a Grid Correction file into the system. When testing installations that contain multiple cards, the file loaded will be associated with the card selected.
These controls allow you to load a Power Table file into the system. When testing installations that contain multiple cards, the file loaded will be associated with the card selected.
Press the GetPosition button to have the program read back the current position. The data is displayed to the left.
Per Helper Card commands
Grid Table
Browse
Set
Reset
Power Table
Browse
Set
Reset
Current Position
0
0
0
GetPosition
Helper Card
Master
Slave1
Slave2
Slave3

Figure 30: Controls that assign correction files to selected HC/3 cards

HC/3 cards have certain parameters related to different types of lasers. For stable operation, CO_2 lasers require tickle pulses to maintain ionization of the gas just below the lasing threshold while the laser is not firing. To enable these pulses, click the check box in the configuration dialog box. The width of the pulses (in microseconds) can be set with the appropriate radio buttons. Pulse train frequency is fixed at 5 kHz. Nd:YAG lasers require a first pulse suppression signal to control the discharge of energy stored in the laser cavity while the laser is not firing. Selection of pulse sense, active high or active low, can be toggled using the “Invert FPS” check box. Extension of the pulse to suppress several pulses may be selected with the radio buttons in the “First Pulse Extension” group.

Once the appropriate settings are made, click the “OK” button. Changes will be implemented the next time the PC is powered on.


HC/3 configuration

- [ ] Inhibit Laser Modulation
- [ ] Enable Tickle Pulses

Tickle Pulse Width

[✓ 0.5]	0.75	1.0	1.25
1.5	1.75	2.0	2.25

FPS Pulse Extension

[✓ No]	1	2	3
4	5	7	8

- [x] Invert FPS
- [x] Separated signals mode
- [x] Invert Mark in Progress

Figure 31: HC/3 configuration dialog box

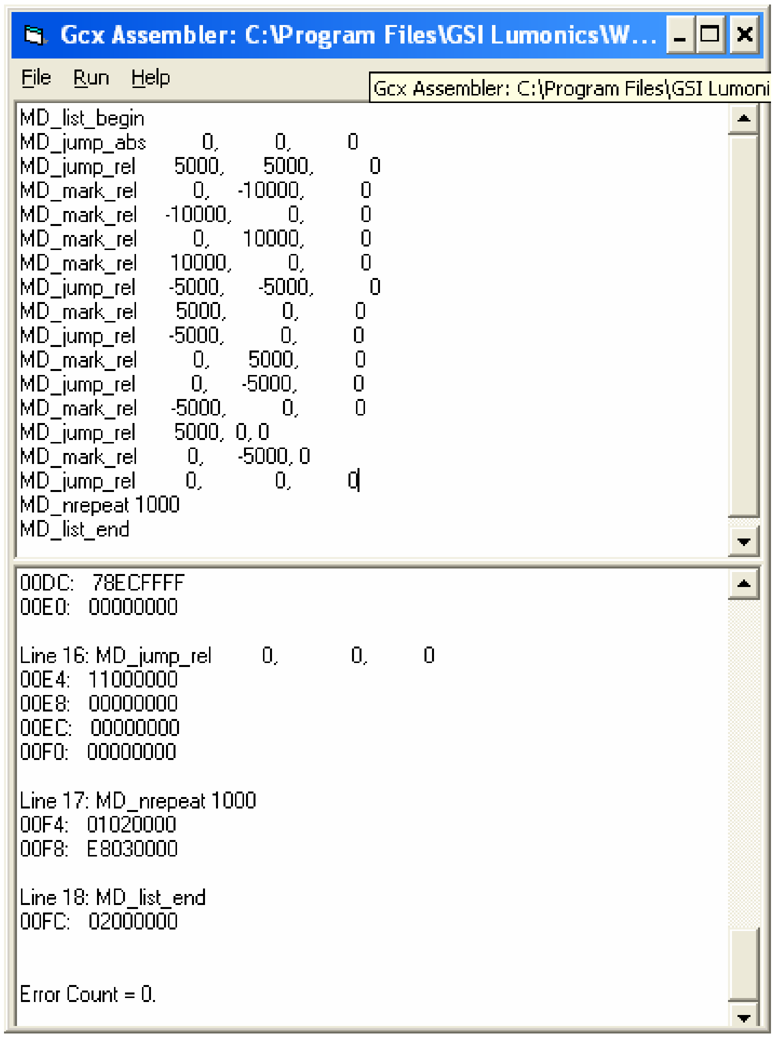
GCX File Assembler Test Program

A utility program GCXAsm.exe is included in the distribution package to generate GCX files. It allows you to type MD commands into a text file, and then assembles those commands into the proper format of a GCX file. In addition, it can disassemble existing GCX files. The following is a summary of menu options

File/New: Create a new GCX source code file and edit its contents in the upper text box
File/Open: If you open an existing GCX source code file then the source code will appear in the upper edit box where it can be edited. If you open an existing GCX file, then it will be disassembled and the disassembled contents will be written to the upper edit box and written to a file of type DIS

File/Save (Save As): Save the edited GCX source code file

Run/Assembler: Assemble the GCX source code file. A GCX and LST fill will be created in the current directory, and the LST file will be displayed in the lower edit box.


Gcx Assembler: C:\Program Files\GSI Lumonics\W...
File
Run
Help
Gcx Assembler: C:\Program Files\GSI Lumoni

```
MD_list_begin
MD_jump_abs      0,      0,      0
MD_jump_rel   5000,   5000,      0
MD_mark_rel      0, -10000,      0
MD_mark_rel -10000,      0,      0
MD_mark_rel      0,  10000,      0
MD_mark_rel  10000,      0,      0
MD_jump_rel  -5000,  -5000,      0
MD_mark_rel   5000,      0,      0
MD_jump_rel  -5000,      0,      0
MD_mark_rel      0,   5000,      0
MD_jump_rel      0,  -5000,      0
MD_mark_rel  -5000,      0,      0
MD_jump_rel   5000, 0, 0
MD_mark_rel      0,   -5000, 0
MD_jump_rel      0,      0,      0
MD_nrepeat 1000
MD_list_end
```

```
00DC:  78ECFFFF
00E0:  00000000

Line 16: MD_jump_rel      0,      0,      0
00E4:  11000000
00E8:  00000000
00EC:  00000000
00F0:  00000000

Line 17: MD_nrepeat 1000
00F4:  01020000
00F8:  E8030000

Line 18: MD_list_end
00FC:  02000000


Error Count = 0.
```

Figure 32: GCXAsm display.

PARAMETER DESCRIPTIONS

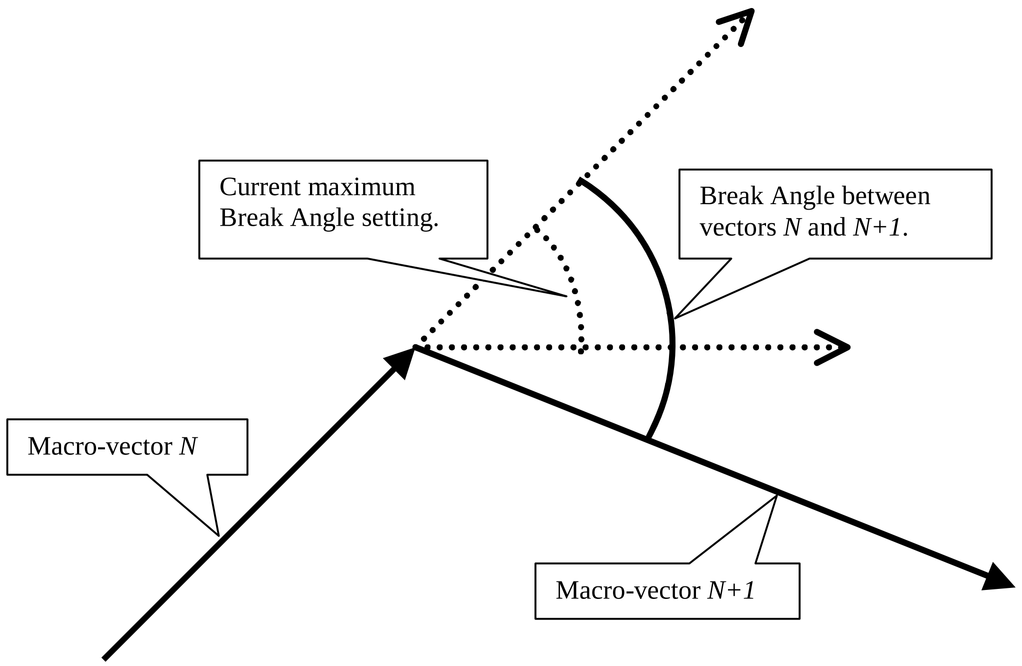
Parameters are used during the execution of GCX files to represent values that seldom change during a mark job. Formerly known as the Laser Parameter Set, the Parameter Set now contains Mark on the Fly velocities as well as parameters that control the quality of the mark on the part. Referencing global parameters, rather than reading them in the GCX instruction stream, will make the execution of GCX instructions more efficient.

The following pages describe the various marking parameters that are stored in the parameter set, click on the parameter below to jump to its description page.

Break Angle

If the angle between two consecutive mark vectors in a stroke is too acute, the beam may not be able accurately to track the change in heading due to galvo/mirror inertia, resulting in a curved error path where "sharp corners" would be expected. We say that the Break Angle, the change in beam "compass heading", is too large.

The *Break Angle* parameter allows a maximum angle to be specified. If this exceeded, WinMCL Plus will break the stroke at the end of vector N, and start a new stroke with vector N+1. If the stroke is broken into two strokes then the laser trajectory executes the *Stroke Delay*. If the included angle is greater than the break angle then the trajectory of the stroke executes without a pause.


Current maximum Break Angle setting.
Break Angle between vectors *N* and *N+1*.
Macro-vector *N*
Macro-vector *N+1*

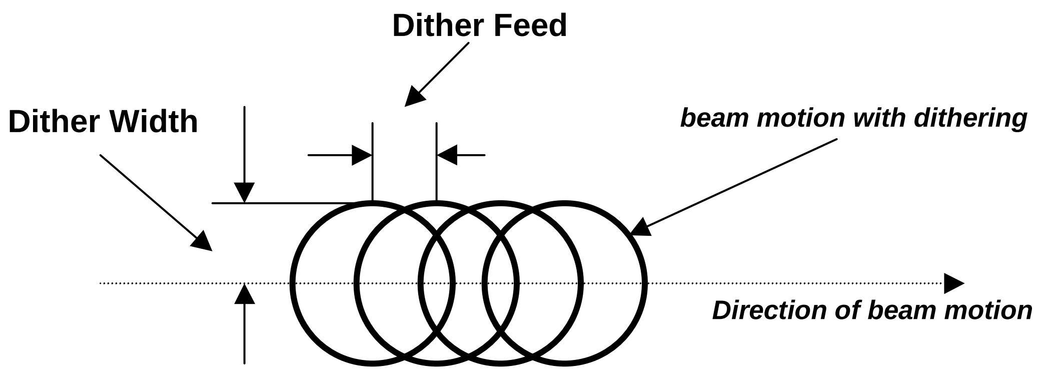
Parameter Definition

Name	Units	Range	Default Value	Type
BreakAngle	Integral Degrees	0 - 180	180	Unsigned long

Related Topics:
Stroke Delay
Polyline Delay
Appendix A: Adjusting Parameters

Dither Feed

The *Dither Feed* parameter is defined as the magnitude of radial variation in the direction of motion of the dither signal. By using the *Dither Feed* and *Dither Width* parameters it is possible to define a variety of circular and oval dither shapes.



Parameter Definition

Name	Units	Range	Default Value	Type
DitherFeed	LSB	Dither Width < DitherFeed < Dither Width* 2	0	Unsigned long

Dither Width (DW) defines the radius of the circles. Dither Feed (DF) is the distance between two circles.
The DF constraint is:

```
DW <= DF <= 2*DW
```

In case of DF = DW the next circle's center is located on the edge of the previous one. In other words, they are 50% overlapped.

In case of DF = 2*DW the circles just touch each other.

Mark Size should be always larger than DW*1.41. The size doesn't make any difference as long as it complies to that rule.

```
DW < MS/SQRT(2)
```

The rate of dither is:

DF per 7*SP (7 is 6 for circle elements + 1 for mark to next circle).

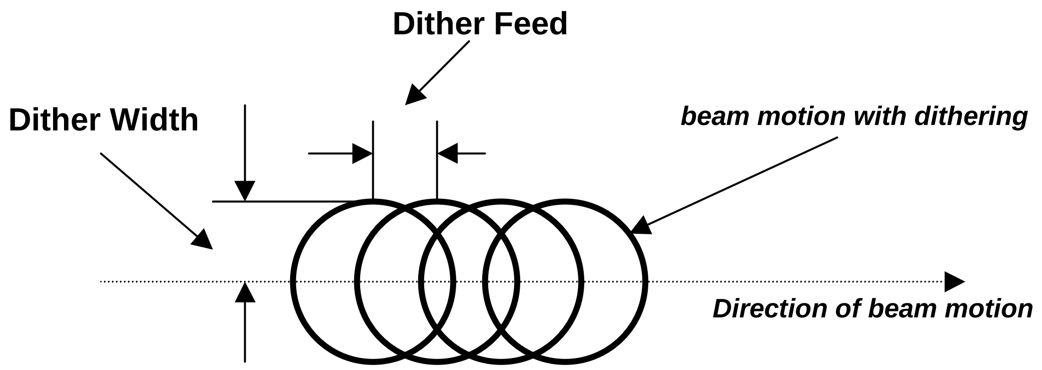
For example, if you have a line of 10000 LSB long and SP = 20 and DF = 50, then the rate is 50 LSB/(7*20 μsec). The line mark time would be 10000*7*20/50 = 28 msec.

Dither Width

The *Dither Width* parameter is defined as the magnitude of radial variation orthogonal to the axis of motion. By using the *Dither Width* and *Dither Feed* parameters it is possible to define a variety of circular and oval dither shapes. If Dither Width is equal 0 then, dither is disabled. If Dither Width is > 0, then dither is enabled. Value constrains for dither width are as follows:

```
DW < MS/SQRT(2)
DW <= DF <= 2*DW
```

where MS is Mark Size, DW and DF are Dither Width and Dither Feed.



Parameter Definition

Name	Units	Range	Default Value	Type
DitherWidth	LSB	DitherWidth <= MarkSize/1.414 && DitherWidth<= DitherFeed<=2*DitherWidth	0	Unsigned long

Related Topics:
Dither Feed
Appendix A: Adjusting Parameters

FPS Enable

The *FPS Enable* parameter enables the use of the First Pulse Suppression signal during laser control. FPS is useful in the control of lasers that build up charge when not lasing, such as lamp or diode pumped YAG lasers.

Parameter Definition

Name	Units	Range	Default Value	Type
FPSEnable	None	0-1	0	Unsigned long

Related Topics:
Laser Power Control
FPS Delay
Appendix A: Adjusting Parameters

Intensity

The *Intensity* parameter specifies the value that will be output from the HC/3's 16-bit output port (Port N).

Parameter Definition

Name	Units	Range	Default Value	Type
Intensity	LSB	0 - 65535	0	Unsigned long

Related Topics:
Hardware IO
Port N output
Appendix A: Adjusting Parameters

Jump Delay

The *Jump Delay* parameter defines the interval of time that the motion trajectory pauses after a jump. This delay is useful for clarity tuning of marks because it defines a time interval where the mirror can settle down after an abrupt motion.

Parameter Definition

Name	Units	Range	Default Value	Type
JumpDelay	µseconds	0 - 10000000	500	Unsigned long

Related Topics:
Fine Tuning with Delays
Appendix A: Adjusting Parameters

Jump Size

The *Jump Size* parameter defines the length of the microvector that is traversed in the *Step Period*. The Jump Size and Step Period parameters define the jump velocity along the beam path.

Parameter Definition

Name	Units	Range	Default Value	Type
JumpSize	LSB	-10000 - 10000	100	Unsigned long

Related Topics:
Jump Rate
Step Period
Appendix A: Adjusting Parameters

Laser OFF Delay

This parameter sets the period of time between the time the laser modulation signal is unasserted and the end of the stroke delay interval.

Parameter Definition

Name	Units	Range	Default Value	Type
LaserOffDelay	µseconds	0 - 10000000	0	Unsigned long

Related Topics:
Fine Tuning with Delays
Appendix A: Adjusting Parameters

Laser ON Delay

This parameter sets the period of time in microseconds between the galvo command that begins a mark and the laser modulation signal assertion. A negative value will assert the laser modulation signal before the command to move the galvo is sent.

Parameter Definition

Name	Units	Range	Default Value	Type
LaserOnDelay	µseconds	0 - 10000000	0	Signed long

Related Topics:
Fine Tuning with Delays
Appendix A: Adjusting Parameters

Mark Size

The *Mark Size* parameter defines the length of the microvector that is traversed in the *Step Period*. The Mark Size and Step Period parameters define the marking velocity along the beam path.

The *Mark Size* parameter is used in the following MD commands:
MD_mark_abs
MD_mark_rel
MD_move_abs
MD_move_rel

Parameter Definition

Name	Units	Range	Default Value	Type
MarkSize	LSB	1 - 10000	100	Unsigned long

Related Topics:
Mark Rate
Step Period
Appendix A: Adjusting Parameters

Polyline Delay

The *Polyline Delay* parameter is used to program a pacing delay that is executed at each internal vertex in a stroke. Please see the tutorial Fine Tuning with Delays.

Parameter Definition

Name	Units	Range	Default Value	Type
PolyLineDelay	µseconds	0 - 10000000	500	Unsigned long

Related Topics:
Fine Tuning with Delays
Appendix A: Adjusting Parameters

Power

The *Power* parameter is used to set the laser power.

Parameter Definition

Name	Units	Range	Default Value	Type
Power	LSB	0-255	0	Unsigned long

Related Topics:
PortB
Appendix A: Adjusting Parameters

Power Delay

The *Power Delay* parameter defines a delay that allows the laser to stabilize when the power is changed. It is executed after any of the following events:

A change in the power table due to either MC_set_powertable or MC_reset_powertable MCL commands.
A Parameter Set change that alters the *Power* parameter

The *Power Delay* is not used in raster mode marking.

Parameter Definition

Name	Units	Range	Default Value	Type
PowerDelay	µseconds	0 - 10000000	1500	Unsigned long

Related Topics:
Power
Appendix A: Adjusting Parameters

Q-Switch Period

The *Q-Switch Period* parameter is used to set the period (1/frequency) of the laser modulation signal. If *Q-Switch Period* = 0 then the laser modulation output is continuous wave (CW).

Parameter Definition

Name	**Units**	**Range**	**Default Value**	**Type**
QSwitchPeriod	µseconds	1 - 1000000 or 0 for CW	100	Unsigned long

Related Topics:
Laser Modulation
Appendix A: Adjusting Parameters

Q-Switch Width

The *Q-Switch Width* parameter is used to set the duty cycle of the laser modulation signal. Duty cycles are computed base upon the Q-Switch Period of the laser modulation signal.

Parameter Definition

Name	Units	Range	Default Value	Type
Q-Switch Width	µseconds	1 - 1000000	75	Unsigned long

Related Topics:
Laser Modulation
Appendix A: Adjusting Parameters

Step Period

The *Step Period* parameter defines the length of time required to for the beam path to traverse either the *Jump Size* or the *Mark Size* distances.

Parameter Definition

Name	Units	Range	Default Value	Type
Step Period	µseconds	10 - 10000	100	Unsigned long

Related Topics:
Jump Rate
Mark Rate
Appendix A: Adjusting Parameters

Stroke Delay

The *Stroke Delay* parameter defines the length of the pause at the end of a stroke.

Parameter Definition

Name	Units	Range	Default Value	Type
StrokeDelay	µseconds	0 - 10000000	500	Unsigned long

Related Topics:
Fine Tuning with Delays
Appendix A: Adjusting Parameters

FUNCTION DEFINITIONS

Various "MC" commands included in the WinMCL Plus library are called by an application program interface (API). A separate subset of "MD" commands, described in Chapter 0: GCX Command Definitions, direct the HC3 for individual marking scripts, or GCX files. Click on the link below to jump to the topic page.

MC_abort_mark

Operational Description:
This function call aborts the currently executing marking job.

When to Call:
This command will terminate the marking job immediately so it is useful as an emergency stop. Note, the beam position and laser command signals will be left in an arbitrary state.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_abort_mark( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Side Effects:
Clears the MARK_IN_PROGRESS signal.

Note:
This command leaves ALL laser signals in an UNDEFINED state.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_close_vector_dump_file

Operational Description:
This function call will close the vector dump file.

When to Call:
Call this function at the appropriate time in your debug routine to halt logging of marking commands to the file.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_close_vector_dump_file( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
MC_open_vector_dump_file

MC_exit

Operational Description:
Shuts down WinMCL32, destroys the software instance and closes the communication channel to the HC/3 device driver.

When to Call:
Make this the last WinMCL Plus call before exiting your program. Once MC_exit is called you will have to reinitialize with MC_init.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_exit();
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
No further commands (with the exception of MC_init) can be successfully issued after MC_exit.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_exit_mark

Operational Description:
Gracefully exit from any repeat or nrepeat loop.

When to Call:
This command is used to exit a repeating mark job. MC_exit_mark is executed asynchronously to the repeating mark job and so may be sensed by the marking thread at an arbitrary point in the job. The action of the command is to prevent the repeat or nrepeat command from repeating, and so one final complete execution of the job is performed. The job queue then starts and the next job loads to be marked. This command is useful in manufacturing situations such as ablation or surface treatment where the host program may be signaled that a desired result has been achieved. The host program would then call MC_exit_mark to halt the processing and move to the next job.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_exit_mark( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_begin_mark

Operational Description:
Gets the status of the external BEGIN_MARK signal on the Master HC/3.

When to Call:
Use this command if the host program needs to read the hardware input of the HC/3 card.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
SignalStatus	The current signal status	Unsigned short *	1 = ASSERTED, 0 = INACTIVE

Function Call:

```
unsigned short SignalStatus;

MC_get_begin_mark( &SignalStatus );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The command should only be issued when WinMCL Plus is idle.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
BEGIN_MARK signal

MC_get_mark_abort

Operational Description:
Gets the status of the external MARK_ABORT signal on the Master HC/3.

When to Call:
Use this command if the host program needs to read the status of the external inputs on the HC/3 card.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
SignalStatus	The current signal status	Unsigned short *	1 = ASSERTED, 0 = INACTIVE

Function Call:

```
unsigned short  SignalStatus;

MC_get_mark_abort( &SignalStatus );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The command should only be issued when WinMCL Plus is idle.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
MARK_ABORT signal

MC_get_optional_status

Operational Description:
Gets the status of the external OPT2 signal on the Master HC/3.

When to Call:
This command is useful if the host program needs to read the status of the external inputs on the HC/3 card.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
SignalStatus	The current signal status	Unsigned short *	1 = ASSERTED, 0 = INACTIVE

Function Call:

```
unsigned short SignalStatus;

MC_get_optional_status( &SignalStatus );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Possible Values for Output Parameter SignalStatus:
1 = ASSERTED
0 = INACTIVE

Note:
The command should only be issued when WinMCL Plus is idle.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
OPT2 signal

MC_get_corrtable

Operational Description:
Gets the Field Distortion Correction Table for the selected HC/3.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters:

Name	Description	Type Declaration	Value Range
CorTbl	The address of the field distortion correction table	CORRTABLE *	0-4294967295

Function Call:

```
unsigned short CardIndex;
CORETABLE * CorTbl;

CorTbl = (CORETABLE *) malloc(sizeof(CORETABLE));
MC_get_corrtable( CardIndex, CorTbl );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Data Structure Definitions

MC_get_cur_xyz

Operational Description:
Gets the current X, Y, and Z output positions for the selected HC/3. The returned position has had field distortion correction applied, and does not necessarily match any particular position in the input GCX data file.

When to Call:
Use this command when you want to know where the laser is pointing. Since the data returned has had the field distortion correction applied it is not trivially possible to understand how the data relates to position data in the GCX file, but it can be useful in certain contexts, such as verifying that the laser is pointing to a safe area.

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters:

Name	Description	Type Declaration	Value Range
XPosition	Current X output position	Long *	-32768 to 32767
YPosition	Current Y output position	Long *	-32768 to 32767
ZPosition	Current Z output position	Long *	-32768 to 32767

Function Call:

```
unsigned short CardIndex;
long Xposition, Yposition, Zposition;

MC_get_cur_xyz( CardIndex, &Xposition, &Yposition, &Zposition );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_error

Operational Description:
Gets the WinMCL Plus error code and message

When to Call:
Each MC function call returns a flag that states if there was in error in processing. These return values should be examined and if there was an error, this function is called in order to retrieve the error code.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
pErrorStruct	A buffer allocated in the calling function or globally, where the fetched error information will be stored.	ERROR_STRUCT *	Previously allocated error structure buffer.

Function Call:

```
ERROR_STRUCT * pErrorStruct;

pErrorStruct = (ERROR_STRUCT *) malloc(sizeof(ERROR_STRUCT));
MC_get_error( pErrorStruct );
```

Possible Return Values:
WMCL_OPERATION_OK

Note:
This command is only useful after a previous command has returned the WMCL_OPERATION_FAILED status.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Error Code Structure

MC_get_fps_delay

Operational Description:
Gets the current first pulse suppression delay value.

When to Call:
Use this call to retrieve the current FPS Delay setting. Useful when you want to make incremental changes to the value.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
FpsDelay	FPS delay in milliseconds	Unsigned long *	Previously allocated unsigned long address

Function Call:

```
unsigned long DelayVal;

MC_get_fps_delay( &DelayVal );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_head_status

Operational Description:
Function gets the head status for the Master HC/3.

When to Call:
Call this command to get data about the operation of the marking head.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Status	Pointer to a memory location in the calling function that will contain the value of the head status byte.	Unsigned short *	Properly allocated unsigned short address.

Function Call:

```
unsigned short stat;

MC_get_head_status( &stat );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Possible Values in Output Parameter Status:

MC_get_head_status places two bytes in Status, with the format

MSB	15	14	13	12	11	10	9	8	7	6	5	4	3	2	LSB
YSR	YT	-	-	XSR	XT	-	-	-	-	-	-	-	-	-	-

Where XSR and YSR are Xand Y servo ready signals. Servo ready "TRUE" is 1.
XT and YT are X and Y temperature out of limits. Temperature OK "TRUE" is 0.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_job_count

Operational Description:
Gets the number of jobs in the job queue.

When to Call:
Use this function to check to see if there are any jobs waiting to be marked. Very useful when two threads are running, one to generate jobs, one to monitor marking. See Job Queue for details.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Status	Pointer to a memory location in the calling function that will contain the value number of jobs in the job queue	Unsigned long *	Properly allocated unsigned long address.

Function Call:

```
unsigned long NumJobs;

MC_get_head_status( &NumJobs );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Job Queue

MC_get_num_cards

Operational Description:
Gets the number of HC/3s detected by the HC/3 device driver.

When to Call:
If the marking program is supposed to support multiple HC/3s (a maximum of 4) then this function is used to determine the number of cards installed in the machine.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
pnCards	Pointer to an unsigned long memory location in the calling function that will contain the number of HC/3 cards installed in the system.	Unsigned long *	Properly allocated unsigned long pointer.

Function Call:

```
unsigned long NumCards

MC_get_num_cards ( &NumCards );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_ParamSet

Operational Description:
Get the values of one of the 64 parameter sets.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
Index	Parameter set index.	int	0 to 63

Output Parameters:

Name	Description	Type Declaration	Value Range
pParamSet	Address of a parameter set structure in the user application that will receive the values of the selected parameter set.	PSET *	Previously allocated parameter set structure address

Function Call:

```
int Index;
PSET * pParamSet;

pParamSet = (PSET *) malloc(sizeof(PSET));
MC_get_ParamSet( Index, pParamSet );
```

Function Call:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Function Call:
General Comments on Function Calls
WinMCL Error Codes
Parameter Set Structure

MC_get_portA_input

Operational Description:
Gets the current state of the Port A input register on a given HC/3.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters:

Name	Description	Type Declaration	Value Range
Value	Current state of Port A	Unsigned short *	Previously allocated unsigned short address

Function Call:

```
unsigned short CardIndex;
long Value;

MC_get_portA_input( CardIndex, &Value );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Port A input

MC_get_powertable

Operational Description:
Gets the Laser Power Calibration Table for the selected HC/3.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters:

Name	Description	Type Declaration	Value Range
Table	address of the power calibration table	POWERTABLE *	Previously allocated POWERTABLE address

Function Call:

```
unsigned short CardIndex;
POWERTABLE * Table;

Table = (POWERTABLE *) malloc(sizeof (POWERTABLE));
MC get powertable( CardIndex, Table );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Data Structure Definitions

MC_get_status

Operational Description:
Get the WinMCL Plus system status. The system status value consists of two groups of bits. The first cumulatively indicates the progress of a job through successive stages of processing. The second indicates error conditions.

When to Call:
This command is used to monitor the progress of the marking thread. It is useful in coordinating the flow of data into the job queue.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Status	The current WinMCL Plus system status code.	Unsigned long *	Previously allocated unsigned long address

Function Call:
```
unsigned long Status;

MC_get_status( &Status );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Possible Values in Output Parameter Status:
STATUS_IDLE
STATUS_LOADING_JOB
STATUS_MARKING
STATUS_WAITING_FOR_BEGIN

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_target_velocity

Operational Description:
Get the target X and Y velocities that were previously set in MC_set_target_velocity. Units are LSB/sec.

When to Call:

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
pXVelocity	Pointer to variable to receive the Target X Velocity. Velocity units are LSB's per second.	float *	Previously allocated float address
pYVelocity	Pointer to variable to receive the Target Y Velocity. Velocity units are LSB's per second.	float *	Previously allocated unsigned float address

Function Call:

```
float pXVelocity, pYVelocity;

MC_get_target_velocity( &pXVelocity, &pYVelocity );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_timeout

Operational Description:
Get the current timeout value (in milliseconds).

When to Call:
Use this function to return the HC/3 watchdog timer setting.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_get_timeout( );
```

Return Value:
The Current Timeout_Value (ms) or 0 if there is an error.

Related Topics:
General Comments on Function Calls

MC_get_version

Operational Description:
Gets the WinMCL Plus version information strings.

When to Call:
This call can be useful in determining if the proper HC/3 device driver is installed on the system. Also, if there is a risk of running older GCX files that contain obsolete commands, this function may allow you to produce a warning.

Input Parameters:

Name	Description	Type Declaration	Value Range
Part	Indicator for which part of the version to be inserted into the buffer. 0=Driver, 1=DLL	Unsigned long	0-1

Output Parameters:

Name	Description	Type Declaration	Value Range
Version	Buffer to hold version string	Char *	Previously allocated buffer 256 bytes long

Function Call:

```
unsigned long Part;
char * Version;

Version = (char *) malloc (256);
MC_get_version( Version, Part );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
For the driver version, Part = WMCLVI_DRIVER and for the DLL, Part = WMCLVI_DLL

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_x_flip

Operational Description:
Get the current x-flip flag value.

When to Call:

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Flag	Current flag state 1 = TRUE, 0 = FALSE	Unsigned short *	Previously allocated unsigned short buffer

Function Call:

```
unsigned short Flag;

MC_get_x_flip( &Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_xform_matrix

Operational Description:
Gets the current transform matrix coefficients for the Universal Matrix. The following diagram shows the row and column locations for each of the output parameters for the Universal matrix. The traditional application of the Universal Matrix is to adjust the overall geometry of the marking system to the part handler.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 33: Placement of the function parameters in the Universal Matrix

When to Call:
This function is useful when minor corrections to the overall marking geometry have to be made. Use this function to capture the existing settings, make the required incremental changes, and then load the new values into the Universal Matrix with the MC_set_xform_matrix.

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
a	Transform matrix coefficient	Float *	Previoulsy allocated float address
b	Transform matrix coefficient	Float *	"
c	Transform matrix coefficient	Float *	"
d	Transform matrix coefficient	Float *	"
x0	Transform matrix coefficient	Float *	"
y0	Transform matrix coefficient	Float *	"

Function Call:
```
float a, b, c, d, x0, y0;

MC_get_xform_matrix( &a, &b, &c, &d, &x0, &y0 );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Marking Field Transformations
Universal Matrix

MC_get_xy_exchange

Operational Description:
Gets the current x-y exchange flag value.

When to Call:

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Flag	Current flag state 1 = TRUE, 0 = FALSE	Unsigned short *	Previoulsy allocated unsigned short address

Function Call:

```
unsigned short Flag;

MC_get_xy_exchange( &Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_get_y_flip

Operational Description:
Get the current y-flip flag value.

When to Call:

Input Parameters: <None>

Output Parameters:

Name	Description	Type Declaration	Value Range
Flag	Current flag state	Unsigned short *	1 = TRUE, 0 = FALSE

Function Call:

```
unsigned short Flag;

MC_get_y_flip( &Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_init

Operational Description:
Instantiates and initializes WinMCL to the default state. It also opens the communication channel to the HC/3 device driver and sets all internal variables and tables (Field Distortion Correction Table and Laser Power Calibration Table) to default values.

When to Call:
Call this function first before any other processing is attempted.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_init( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
Must be called once only before any other commands are issued.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Laser Power Calibration Table default value

MC_jump_abs

Operational Description:
Performs a controlled JUMP ABS to the given location, using the current laser parameter set.

When to Call:
This command is useful when the marking jobs consist solely of relative motion commands. The host program uses this command to jump to the origin of a part, and then the marking routine is loaded and called. This can also be a step and repeat operation, where the jump command is used to slew to the beginning of the next repeat.

Input Parameters:

Name	Description	Type Declaration	Value Range
X	X coordinate	Long	-32768 to 32767
Y	Y coordinate	Long	-32768 to 32767
Z	Z coordinate	Long	-32768 to 32767

Output Parameters: <None>

Function Call:

```
long X, Y, Z ;

MC_jump_abs( X, Y, Z );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_jump_rel

Operational Description:
Perform a controlled JUMP REL to the given location, using the current laser parameter set.

When to Call:
This command is useful in step-and-repeat applications where the marking program consists solely of relative motion commands.

Input Parameters:

Name	Description	Type Declaration	Value Range
X	X offset	Long	-32768 to 32767
Y	Y offset	Long	-32768 to 32767
Z	Z offset	Long	-32768 to 32767

Output Parameters: <None>

Function Call:

```
long X, Y, Z;

MC_jump_rel( X, Y, Z );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_load_job

Operational Description:
Opens a GCX File located on disk and places it in the Job Queue for subsequent processing.

When to Call:
This particular command assumes that the marking job is described in a GCX File in the files system. So this command reads and parses the file and then pushes the job onto the job queue.

Input Parameters:

Name	Description	Type Declaration	Value Range
GCX_filename	Full path file name expressed in an even number of unicode characters.	Char *	Valid file name.

Output Parameters: <None>

Function Call:

```
char * GCX_filename;

GCX_filename = (char *) malloc(512);
MC_load_job( GCX_filename );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_load_job_from_memory

Operational Description:
Opens GCX Data in application memory and places it in the Job Queue for subsequent processing.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
startAddress	Pointer to the start of the job file in application memory.	Char *	Properly allocated block of memory
Length	Length of the job file (in bytes).	Unsigned long	0-4294967295

Output Parameters: <None>

Function Call:

```
char *startAddress;
unsigned long length;

startAddress = (char *) malloc(512);
MC_load_job_from_memory( startAddress, length );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_open_vector_dump_file

Operational Description:
Opens the vector dump file.

When to Call:
This is a function used in debugging marking jobs. The vector dump file will contain the binary representations of all the MD commands of a marking job.

Input Parameters:

Name	Description	Type Declaration	Value Range
pszFileName	Name of file.	Char *	Previously allocated char buffer at least 512 bytes long

Output Parameters: **<None>**

Function Call:

```
char * pszFileName;

pszFileName = (char *) malloc(512);
MC_open_vector_dump_file( pszFileName );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_reset

Operational Description:
Resets WinMCL Plus to the default state. This affects all Global Parameters, Parameter Sets and tables (Field Distortion Correction Table and Laser Power Calibration Table).

When to Call:
Call this function to force a return of all tables and variables to a known state.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_reset( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Side Effects:
Clears the MARK_IN_PROGRESS signal.
Clear the MARK_ERROR signal

Note:
The communication channel to the kernel mode driver is not affected by this command. In all other respects, MC_reset is equivalent to MC_init.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Laser Power Calibration Table default value
MARK_IN_PROGRESS

MC_reset_corrtable

Operational Description:
Resets the Field Distortion Correction Table for the selected HC/3 to the default values (i.e. no corrections).

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters: <None>

Function Call:

```
unsigned short CardIndex;

MC_reset_corrtable( CardIndex );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_reset_job_queue

Operational Description:
Resets the Job Queue

When to Call:

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_reset_job_queue();
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_reset_ParamSet

Operational Description:
Reset the values of one of the 64 parameter sets to default values.

When to Call:
This function is useful when you want to start with a parameter set that has known values. Call this function instead of setting the value of each of the parameters in the set.

Input Parameters:

Name	Description	Type Declaration	Value Range
Index	Parameter set index	int	0 to 63

Output Parameters: <None>

Function Call:

```
int Index;

MC_reset_ParamSet( Index );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Parameter Set Structure

MC_reset_powertable

Operational Description:
Resets the Laser Power Calibration Table for the selected HC/3 to the default values (i.e. 1:1 power mapping)

When to Call:
This function is useful when you want to perform alter data in the power table, but you want the power table to be in a known state.

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3

Output Parameters: <None>

Function Call:

```
unsigned short CardIndex;

MC_reset_powertable( CardIndex );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_reset_tracking

Operational Description:
Reset the mark on the fly tracking variables.

When to Call:
This function should be called after the marking operation is complete and WinMCL is idle. The mark on the fly tracking counter can only hold a finite count, so after each MOF mark operation the counters should be reset so that they do not roll over.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_reset_tracking( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Mark on the Fly

MC_select_parameter_set

Operational Description:
Selects a laser parameter set as the current parameter set.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
Index	Parameter Set Index	Long	0 - 63

Output Parameters: <None>

Function Call:

```
long Index;

MC_select_parameter_set( Index );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Parameter Set Structure

MC_set_mark_error

Operational Description:
Set or reset the external MARK_ERROR signal on all HC/3s.

When to Call:
Call if you wish to change the state of pin 9 of the External I/O connector.

Input Parameters:

Name	Description	Type Declaration	Value Range
SignalState	The required signal state	Unsigned short	1 = ASSERTED, 0 = INACTIVE

Output Parameters: <None>

Function Call:

```
unsigned short SignalState;

MC_set_mark_error( SignalState );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The command should only be issued when WinMCL Plus is idle.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
MARK_ERROR signal

MC_set_remote_execute

Operational Description:
Set or reset the external REMOTE_EXECUTE signal on all HC/3s.

When to Call:

Call when you wish to change the state of pin 8 of the External I/O connector.

Input Parameters:

Name	Description	Type Declaration	Value Range
SignalState	The required signal state	Unsigned short	1 = ASSERTED, 0 = INACTIVE

Output Parameters: <None>

Function Call:

```
unsigned short SignalState;

MC_set_remote_execute( SignalState );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The command should only be issued when WinMCL Plus is idle.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
REMOTE_EXECUTE signal

MC_set_corrtable

Operational Description:
Loads the Field Distortion Correction Table for the selected HC/3 from the given file.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3
FileName	Field distortion correction table file name	Char *	Must an allocated string that contains the file name.

Output Parameters: **<None>**

Function Call:

```
unsigned short CardIndex;
char * FileName;

FileName = (char *) malloc (512);
MC_set_corrtable( CardIndex, FileName );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_corrtableFromMemory

Operational Description:
Copies correction table values to the indexed correction table from memory.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
DestIndex	Card index of the destination correction table.	Unsigned short	0 - 3
SourceCorrectionTable	Address of the source correction table.	CORRTABLE *	Must be CORRTABLE * previously allocated.

Output Parameters: **<None>**

Function Call:

```
unsigned short DestIndex;
CORRTABLE * SourceCorrectionTable;

SourceCorrectionTable = (CORRTABLE *) malloc(sizeof(CORRTABLE));
MC_set_corrtableFromMemory( DestIndex,
SourceCorrectionTable );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_external_signal

Operational Description:
Set the external signal (pin 50 of the I3 connector). This represents the time that should elapse between the FPS Trigger Pulse and the start of Laser Modulation

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
State	On/Off	unsigned short	0 / 1

Output Parameters:

Function Call:

```
unsigned short usOnOff = 1;

MC_set_external_signal(State);
```

Possible Return Values:

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_fps_delay

Operational Description:
Set the first pulse suppression delay value. This represents the time that should elapse between the FPS Trigger Pulse and the start of Laser Modulation.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
FpsDelay	FPS delay in microseconds	Unsigned long	0-4294967295

Output Parameters: **<None>**

Function Call:

```
unsigned long FpsDelay;

MC_set_fps_delay( FpsDelay );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_ParamSet

Operational Description:
Set the values of one of the 64 parameter sets.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
Index	Parameter set index.	int	0 to 63
pParamSet	Address of a parameter set structure in the user application that contains the values to write to the selected parameter set.	PSET *	Must be PSET * previously allocated

Output Parameters: <None>

Function Call:

```
int Index;
PSET * pParamSet;

pParamSet = (PSET *)malloc(sizeof(PSET));
MC_set_ParamSet( pParamSet );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Parameter Set Structure

MC_set_powertable

Operational Description:
Loads the Laser Power Calibration Table for the selected HC/3 from the given file.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3
FileName	Power calibration table file name	Char *	Previouly allocated char buffer at least 512 bytes long

Output Parameters: <None>

Function Call:

```
unsigned short CardIndex;
char * FileName;

CardIndex = (char *) malloc(512);
MC_set_powertable( CardIndex, FileName );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The table describes the mapping between the power values given in MD_set_power data records and the corresponding laser interface values.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_powertableFromMemory

Operational Description:
Loads the Laser Power Calibration Table for the selected HC/3 from application memory.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
CardIndex	HC/3 number	Unsigned short	0 - 3
ptptr	Pointer to power table in memory	POWERTABLE *	Pointer to previously allocated and populated power table.

Output Parameters: <None>

Function Call:

```
unsigned short CardIndex;
POWERTABLE * ptptr;

ptptr = (POWERTABLE *) malloc(sizeof(POWERTABLE));
MC_set_powertableFromMemory ( CardIndex, prptr );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:
The table describes the mapping between the power values given in MD_set_power data records and the corresponding laser interface values.

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_raster_mode

Operational Description:
This function is used to set the type of laser control that is output when either the MD_raster_abs or MD_raster_rel commands are executed during a marking job. There are three mode of Raster Pulse control.

When to Call:
This is a configuration command used to select the laser control option for generating Raster Pulses.

Input Parameters:

Name	Description	Type Declaration	Value Range	
			Value	**Action**
RasterMode	Raster mode	Unsigned long	0	PortB and LM
			RASTER_8BIT_ONLY	PortB only
			RASTER_PWM_ONLY	LM only

Output Parameters: <None>

Function Call:

```
unsigned long RasterMode;

MC_set_raster_mode( RasterMode );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_target_velocity

Operational Description:
Set the target X and Y velocities.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
XVelocity	Target X Velocity. Velocity units are LSB's per second.	Float	Float limits
YVelocity	Target Y Velocity. Velocity units are LSB's per second.	Float	Float limits

Output Parameters: <None>

Function Call:

```
float Xvelocity, YVelocity;

MC_set_target_velocity( Xvelocity, YVelocity );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Mark on the Fly

MC_set_timeout

Operational Description:
Specifies how long WinMCL Plus should wait for a response from the HC/3 device driver. The default value (after MC_init or MC_reset) is 10 seconds, which is more than adequate in all normal circumstances. The value may need to be increased if using a very long Step Periods (roughly, more than 5 milliseconds, although this value is highly dependent upon the exact mixture of vectors produced by the application).

When to Call:
While this function is provided it is not necessary for normal operation. The length of time that the HC/3 card is away processing is dependent on the complexity of the marking job and the Mark and Jump rates. If it is absolutely required that the host program set a watchdog over the HC/3 process then use this function to set an upper limit on the time that any single process can complete.

Input Parameters:

Name	Description	Type Declaration	Value Range
NewTime	The desired timeout value in milliseconds	Unsigned long	0-4294967295

Output Parameters: <None>

Function Call:

```
unsigned long NewTime;

MC_set_timeout( NewTime );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_x_flip

Operational Description:
Set or reset the x-flip flag. When the flag is TRUE, the command stream output by the HC/3 card will have the X-axis reverse with respect to the GCX command file. X-Flip does not modify the Transform Matrix.

When to Call:
This function can be called during initialization of the system to compensate for mounting geometries or it can be called in response to input data about the orientation of the part in the field of view.

Input Parameters:

Name	Description	Type Declaration	Value Range
Flag	Current flag state	Unsigned short	1 = TRUE, 0 = FALSE

Output Parameters: <None>

Function Call:

```
unsigned short Flag;

MC_set_x_flip( Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_xform_matrix

Operational Description:
Sets the transform matrix coefficients for the Universal Matrix. The transform matrix is applied to all incoming vectors to provide scaling, rotation, and other transformations of the vector coordinates before field distortion correction is applied. The following diagram shows the row and column locations for each of the input parameters for the Universal matrix.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
a	Transform matrix coefficient value	Float	Float limits
b	Transform matrix coefficient value	Float	Float limits
c	Transform matrix coefficient value	Float	Float limits
d	Transform matrix coefficient value	Float	Float limits
x0	Transform matrix coefficient value	Float	Float limits
y0	Transform matrix coefficient value	Float	Float limits

Output Parameters: <None>

Function Call:

```
float a, b, c, d, x0, Y0;

MC_set_xform_matrix( a, b, c, d, x0, y0 );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Marking Field Transformations
Universal Matrix

MC_set_xy_exchange

Operational Description:
Set or reset the x-y exchange flag. When the flag is TRUE, the X and Y output channels will be swapped in relation to the commands in the GCX file. XY_exchange does not affect the Transform Matrix.

When to Call:

Input Parameters:

Name	Description	Type Declaration	Value Range
Flag	Desired flag state	Unsigned short	1 = TRUE, 0 = FALSE

Output Parameters: <None>

Function Call:

```
unsigned short Flag;

MC_set_xy_exchange( Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_set_y_flip

Operational Description:
Set or reset the y-flip flag. When the flag is TRUE, the command stream output by the HC/3 card will have the Y-axis inverted with respect to the GCX command file. Y-Flip does not modify the Transform Matrix.

When to Call:
This function can be called during initialization of the system to compensate for mounting geometries or it can be called in response to input data about the orientation of the part in the field of view.

Input Parameters:

Name	Description	Type Declaration	Value Range
Flag	Current flag state	Unsigned short	1 = TRUE, 0 = FALSE

Output Parameters: **<None>**

Function Call:

```
unsigned short Flag;

MC_set_y_flip( Flag );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_shoot

Operational Description:
This command turns the laser ON for a time interval. The time interval is programmed by the Shoot_Time parameter of the function.

When to Call:
This command is typically used in "via hole" drilling applications, a step in the production of printed circuit boards. When used in conjunction with the MC_jump_rel and MC_jump_abs commands it is possible to develop a via hole drilling application that avoids the use of GCX files. The program consists of jump commands that set the focal point to the appropriate location and the MC_shoot command that operates the laser.

It is not appropriate to use the MC_shoot command in vector or raster marking operations and if the MARK_IN_PROGRESS signal is asserted while the MC_shoot command is issued, an error will be generated and the marking operation will be aborted.

Input Parameters:

Name	Description	Type Declaration	Value Range
ShootTime	When using PWM, the n parameter of the MD_shoot command defines the number of pulses to be fired. In case of continuous PWM output (PWM period = 0), n defines the shoot time in microseconds.	Unsigned long	Float limits

Output Parameters: <None>

Function Call:

```
unsigned long ShootTime;

MC_shoot( ShootTime );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_start_mark

Operational Description:
Commence the execution of GCX marking jobs that have been loaded into the job queue. This call will not return until the job queue is empty.

When to Call:
Once the job queue contains marking jobs they can be marked. Use this command to commence marking.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_start_mark( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_start_mark_non_blocking

Operational Description:
As above, except this function spawns its own thread and returns essentially immediately. Use with languages like LabVIEW that have difficulty spawning their own threads.

When to Call:
Once the job queue contains marking jobs they can be marked. Use this command to commence marking.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_start_mark_non_blocking( );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes

MC_start_mark_on_begin

Operational Description:
Initializes vector processing, and begins vector processing and output. The output starts when the external BEGIN_MARK signal is asserted on the 9-pin Sub-D connector. The MARK_IN_PROGRESS signal is set and the HC/3 starts data output automatically. The command effectively blocks while waiting for the BEGIN_MARK signal. Once the signal has been asserted, the command starts a new thread to run the mark, and returns to the caller.

When to Call:
This command is used once jobs are loaded in the job queue. When the host software issues this command it relinquishes control of the actual start time to the hardware input BEGIN_MARK signal on the HC/3. This is useful when coordinating the operation of manufacturing equipment.

Input Parameters:

Name	Description	Type Declaration	Value Range
TimeOut	Maximum time, in milliseconds, that the program will wait for the "START_MARK" signal.	Unsigned long	0-4294967295

Output Parameters: <None>

Function Call:

```
unsigned long TimeOut;

MC start mark on begin( TimeOut );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
BEGIN_MARK signal

MC_start_mark_on_begin_non_blocking

Operational Description:
As above, except this function spawns its own thread and returns essentially immediately.   Use with languages like LabVIEW that have difficulty spawning their own threads.

When to Call:
This command is used once jobs are loaded in the job queue. When the host software issues this command it relinquishes control of the actual start time to the hardware input BEGIN_MARK signal on the HC/3. This is useful when coordinating the operation of manufacturing equipment.

Input Parameters:

Name	Description	Type Declaration	Value Range
TimeOut	Maximum time, in milliseconds, that the program will wait for the "START_MARK" signal.	Unsigned long	0-4294967295

Output Parameters: <None>

Function Call:

```
unsigned long TimeOut;

MC start mark on begin( TimeOut );
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Note:

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
BEGIN_MARK signal

MC_start_tracking

Operational Description:
This function will enable the mark on the fly capability of WinMCL Plus.

When to Call:
Normally, the part to be marked will be moving along an assembly line. A position sensor along the line will trigger when the part moves into the field of view of the marking head. This trigger can be input into the host program, through one of the input ports of the HC/3 card perhaps, and the code branches on the trigger event to execute this instruction and commence the appropriate marking job.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_start_tracking();
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Mark on the Fly
MC_reset_tracking
MD_start_tracking
MD_reset_tracking

MC_stop_tracking

Operational Description:
This function will enable the mark on the fly capability of WinMCL Plus.

When to Call:
Call this function after the part has been marked.

Input Parameters: <None>

Output Parameters: <None>

Function Call:

```
MC_stop_tracking();
```

Possible Return Values:
WMCL_OPERATION_OK
WMCL_OPERATION_FAILED

Related Topics:
General Comments on Function Calls
WinMCL Error Codes
Mark on the Fly
MC_reset_tracking
MD_start_tracking
MD_reset_tracking

GCX COMMAND DEFINITIONS

WinMCL Plus includes a subset of "MD" commands used to write marking scripts, typically called from GCX files. Click on any of the commands below to jump to the topic page.

MD_comment_begin

The MD_comment_begin record marks the start of a comment block. Comments must consist of an EVEN number of arbitrary Unicode characters (so that the entire record is an integer multiple of DWORDs long) in other words, the string length has to be a multiple of four. A string that actually has an odd number of characters should be padded with a trailing Unicode-NUL.

Record Layout:	<MD_comment_begin>
Opcode:	0x00000007
Record Size:	4 Bytes
Parameters:	None
Restrictions:	Must be paired with an MD_comment_end record. May not be used recursively within comments.

Related Topics:
GCX file tutorial

MD_comment_end

In the syntax of the GCX file this command acts as the end of comment delimiter. Any comments started in the GCX file (using the MD_comment_begin command) must be terminated with this command.

Record Layout:	<MD_comment_end>
Opcode:	0x00000000
Record Size:	4 Bytes
Parameters:	None
Restrictions:	Must be paired with an MD_comment_begin command.

Related Topics:
GCX file tutorial

MD_global_matrix

Specify coefficients for the Global Matrix.

Record Layout:	<MD_global_matrix> <float a, float b, float c, float d, float x0, float y0>
Opcode:	0x00000023
Record Size:	4 + 4 + 4 + 4 + 4 + 4 + 4 = 28 Bytes
Parameters:	a, b, c, d, x0, y0 : Transform Matrix coefficients
Restrictions:	None

The following diagram shows the row and column locations for each of the input parameters for the global matrix.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_global_transform

MD_global_transform

Specify coefficients for the Global Matrix, using alternate parameters. The operation is performed as rotate, then translate.

Record Layout:	<MD_global_transform> <float ang, float x, float y >
Opcode:	0x00000020
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	ang : angle in Radians x : offset in LSB y : offset in LSB
Restrictions:	None

The following matrix operation is performed with the input parameters to generate the global matrix.

$$\begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_global_matrix

MD_include

Include contents of another GCX file.

Record Layout:	<MD_include> <filename>
Opcode:	0x00000005
Record Size:	4 + { 4 ... 512 } Bytes
Parameters:	filename : The filename parameter must be a valid Windows filename, including the extension, of up to 256 UNICODE characters, and must be an EVEN number of Unicode characters long (so that the entire record is an integer multiple of DWORDs long). A name that actually has an odd number of characters should be padded with a trailing Unicode-NUL.
Restrictions:	Must be paired with an MD_include_end record.

Related Topics:
GCX file tutorial

MD_include_data

Identifies GCX data to be inserted contiguously into the current stream. If it occurs outside a Data List Block, it identifies a GCX-File structure (in memory). If it occurs within a Data List Block, it identifies another Data List Block.

Record Layout:	<MD_include_data> <pvoid pGCXdata> <uint length>
Opcode:	0x00000006
Record Size:	4 + 4 + 4 = 12 Bytes
Parameters:	pGCXData : address of start of data. length : of data (bytes)
Restrictions:	None

Related Topics:
GCX file tutorial

MD_include_end

Marks the end of an include block.

Record Layout:	<MD_include_end>
Opcode:	0x00000000
Record Size:	4 Bytes
Parameters:	none
Restrictions:	Must be paired with an MD_include record.

Related Topics:
GCX file tutorial

MD_jump_abs

Use the MD_jump_abs command to move the mirrors to an absolute coordinate, at the jump velocity. During the move the laser will be OFF. The command uses the Jump Size and Step Period parameters from the currently selected Parameter Set to determine the jump velocity.

Record Layout:	<MD_jump_abs> <int x_abs> <int y_abs> <int z_abs>
Opcode:	0x00000010
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_abs : X-axis coordinate y_abs : Y-axis coordinate z_abs : Z-axis coordinate
Restrictions:	None

Related Topics:
GCX file tutorial

MD_jump_rel

Use the MD_jump_rel command to move the mirrors to a new location relative to the current location as specified by relative coordinates, at the jump velocity. During the move the laser will be OFF. The command uses the Jump Size and Step Period parameters from the currently selected Parameter Set to determine the jump rate.

Record Layout:	<MD_jump_rel> <int x_rel> <int y_rel> <int z_rel>
Opcode:	0x00000011
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_rel : X-axis relative coordinate y_rel : Y-axis relative coordinate z_rel : Z-axis relative coordinate
Restrictions:	None

Related Topics:
GCX file tutorial

MD_list_begin

Marks the start of a Data List Block.

Record Layout:	<MD_list_begin>
Opcode:	0x00000001
Record Size:	4 Bytes
Parameters:	None
Restrictions:	Must be paired with an MD_list_end record. May not be used recursively within a List Block.

Related Topics:
GCX file tutorial

MD_list_end

Marks the end of a Data List Block.

Record Layout:	<MD_list_end>
Opcode:	0x00000002
Record Size:	4 Bytes
Parameters:	None
Restrictions:	Must be paired with an MD_list_begin record.

Related Topics:
GCX file tutorial

MD_mark_abs

Use the MD_mark_abs to mark a vector using absolute coordinates, at the marking velocity. While the mirrors are moving the laser will be ON. The command uses the Mark Size and Step Period parameters from the currently selected Parameter Set to determine the marking rate.

Record Layout:	<MD_mark_abs> <int x_abs> <int y_abs> <int z_abs>
Opcode:	0x00000012
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_abs : X-axis coordinate y_abs : Y-axis coordinate z_abs : Z-axis coordinate
Restrictions:	None

Related Topics:
GCX file tutorial

MD_mark_rel

Use the MD_mark_rel command to mark a vector using relative coordinates, at the marking velocity. While the mirrors are moving the laser will be ON. The command uses the Mark Size and Step Period parameters from the currently selected Parameter Set to determine to determine the marking velocity.

Record Layout:	<MD_mark_rel> <int x_rel> <int y_rel> <int z_rel>
Opcode:	0x00000013
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_rel : X-axis relative coordinate y_rel : Y-axis relative coordinate z_rel : Z-axis relative coordinate
Restrictions:	Current position must have been previously established using absolute coordinates.

Related Topics:
GCX file tutorial

MD_move_abs

Use the MD_move_abs to move the mirrors to a location specified by absolute coordinates, at the marking velocity. During the move the laser will be OFF. The command uses the Mark Size and Step Period parameters from the currently selected Parameter Set to determine the marking velocity.

Record Layout:	<MD_move_abs> <int x_abs> <int y_abs> <int z_abs>
Opcode:	0x00000014
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_abs : X-axis coordinate y_abs : Y-axis coordinate z_abs : Z-axis coordinate
Restrictions:	None

Related Topics:
GCX file tutorial

MD_move_rel

Use the MD_move_rel command to move the mirrors to a new location relative to the current location as specified by relative coordinates, at the marking velocity. During the move the laser will be OFF. The command uses the Mark Size and Step Period parameters from the currently selected Parameter Set to determine the marking velocity.

Record Layout:	<MD_move_rel> <int x_rel> <int y_rel> <int z_rel>
Opcode:	0x00000015
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	x_rel : X-axis relative coordinate y_rel : Y-axis relative coordinate z_rel : Z-axis relative coordinate
Restrictions:	Current position must have been previously established using absolute coordinates.

Related Topics:
GCX file tutorial

MD_nrepeat

Use this command to repeat a list <count> number of times.

Record Layout:	< MD_nrepeat > <int count>
Opcode:	0x00000201
Record Size:	8 Bytes
Parameters:	count : number of times to repeat
Restrictions:	None

Related Topics:
GCX file tutorial

MD_object_transform

Specify coefficients for the Object Matrix, using rotation parameters. The operation is performed as translate, then rotate.

Record Layout:	<MD_object_transform> < float ang, float x, float y>
Opcode:	0x00000022
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	ang : angle in Radians x : offset in LSB y : offset in LSB
Restrictions:	None

The following matrix operation is performed with the input parameters to generate the object matrix.

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_object_matrix

MD_object_matrix

Specify coefficients for the Object Matrix.

Record Layout:	<MD_object_matrix> <float a, float b, float c, float d, float x0, float y0>
Opcode:	0x00000025
Record Size:	4 + 4 + 4 + 4 + 4 + 4 + 4 = 28 Bytes
Parameters:	a,b,c,d,x0,y0 Transform Matrix coefficients
Restrictions:	None

The following diagram shows the row and column locations for each of the input parameters for the object matrix.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_object_transform

MD_part_matrix

Specify coefficients for the Part Matrix.

Record Layout:	<MD_part_matrix> <float a, float b, float c, float d, float x0, float y0>
Opcode:	0x00000024
Record Size:	4 + 4 + 4 + 4 + 4 + 4 + 4 = 28 Bytes
Parameters:	a,b,c,d,x0,y0 Transform Matrix coefficients
Restrictions:	None

The following diagram shows the row and column locations for each of the input parameters for the part matrix.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_part_transform

MD_part_transform

Specify coefficients for the Part Matrix, using alternate parameters. The operation is performed as translate, then rotate.

Record Layout:	<MD_part_transform> <float ang, float x, float y >
Opcode:	0x00000021
Record Size:	4 + 4 + 4 + 4 = 16 Bytes
Parameters:	ang : angle in Radians x : offset in LSB y : offset in LSB
Restrictions:	None

The following matrix operation is performed with the input parameters to generate the part matrix.

$$\begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Related Topics:
GCX file tutorial
Marking Field Transformations
MD_part_matrix

MD_raster_abs

This command is used to mark a line of dots on the part. The line is marked from the current position to the end point, where the endpoint is expressed by the <X> and <Y> parameters in absolute coordinates. The next parameter, <n> gives the number of dots that will be marked and what follows are <n> parameters, the gray scale (GSV) value of each dot. Please see the section on raster mode marking for a complete explanation.

Record Layout:	< MD_raster_abs > <X> <Y> <n> <g(0)> <g(1)> <…> <g(n-1)>
Opcode:	0x00000210
Record Size:	4+4+4+4n = 12 + 4n Bytes
Parameters:	X: X-axis endpoint in absolute coordinates Y: Y-axis endpoint in absolute coordinates n: number of pixels in the line g(0) - g(n-1) : array of gray scale values, one for each pixel.
Restrictions:	Selection of raster mode has to be executed before the job is run. See the MC_set_raster_mode function. The following parameters are involved in raster operations: Q-Switch Period, Laser_on_delay, laser_off_delay.

Related Topics:
GCX file tutorial
raster mode tutorial
RasterPulses

MD_raster_rel

This command is used to mark a line of dots on the part. The line is marked from the current position to the end point, where the endpoint is expressed by the <X> and <Y> parameters in coordinates relative to the current position. The next parameter, <n> gives the number of dots that will be marked and what follows are <n> parameters, the gray scale value (GSV) of each dot. Please see the section on raster mode marking for a complete explanation.

Record Layout:	< MD_raster_rel > <X> <Y> <n> <g(0)> <g(1)> <…> <g(n-1)>
Opcode:	0x00000211
Record Size:	4+4+4+4n = 12 + 4n Bytes
Parameters:	X: X-axis endpoint in relative coordinates Y: Y-axis endpoint in relative coordinates n: number of pixels in the line g(0) - g(n-1) : array of gray scale values, one for each pixel.
Restrictions:	Selection of raster mode has to be executed before the job is run. See the MC_set_raster_mode function. The following parameters are involved in raster operations: Q-Switch Period, Laser_on_delay, laser_off_delay.

Related Topics:
GCX file tutorial
raster mode tutorial
RasterPulses

MD_repeat

Use this command to repeat a list of MD commands indefinitely. The marking job must be terminated with with the MC_exit_mark command issued from a separate thread of the host program.

Record Layout:	< MD_repeat >
Opcode:	0x00000200
Record Size:	4 Bytes
Parameters:	None
Restrictions:	None

Related Topics:
GCX file tutorial

MD_reset_tracking

This command resets the Mark on the Fly timing registers to zero.

Record Layout:	< MD_reset_tracking >
Opcode:	0x00000202
Record Size:	4 Bytes
Parameters:	None
Restrictions:	None

Related Topics:
GCX file tutorial
Mark on the Fly tutorial

MD_select_parameter_set

Select active parameter set for use during subsequent vector processing.

Record Layout:	<MD_select_parameter_set> <unit index>
Opcode:	0x0000000A
Record Size:	4 + 4 = 8 Bytes
Parameters:	index parameter set index { 0 ... 63 }
Restrictions:	None

Related Topics:
GCX file tutorial
Parameter Set Structure

MD_set_mark_error

Use this command to change the value of the mark_error signal on pin 9 of the External I/O connector.. The parameter <state> determines the state of the output: A parameter value of 1 asserts the signal, a parameter value of 0 unasserts the signal.

Record Layout:	< MD_set_mark_error > <int state>
Opcode:	0x00000205
Record Size:	8 Bytes
Parameters:	state : On = 1, Off = 0
Restrictions:	None

Related Topics:
GCX file tutorial
Hardware IO

MD_set_remote_execute

Use this command to change the state of the remote_execute signal on pin 8 of the External I/O connector. The <state> parameter determines the state of the signal. A parameter value of 1 asserts the signal, a parameter value of 0 unasserts the signal.

Record Layout:	< MD_set_remote_execute > <int state>
Opcode:	0x00000204
Record Size:	8 Bytes
Parameters:	state: On = 1, Off = 0
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_output_offset

Identifies the offset to be applied to the data for the specified scan head. The offset will be scaled through the universal matrix. This will reset the beam dump output mode for the specified head.

Record Layout:	< MD_Set_Output_Offset > <uint CardIndex > <int Xoffset> <int Yoffset> <int Zoffset>
Opcode:	0x00000026
Record Size:	4 + 4 + 4 + 4 + 4 = 20 Bytes
Parameters:	CardIndex: HC/3 card to apply offset to Xoffset: the X-axis offset Yoffset: the Y-axis offset Zoffset: the Z-axis offset
Restrictions:	None

Related Topics:
GCX file tutorial
MD_Set_BeamDump_Position

MD_shoot

Use the MD_shoot command to turn laser ON for an interval. During the laser ON cycle the focal point will not be moving. The ShootTime parameter is interpreted in two different ways, depending on the configuration of the Laser Modulation Signal. If the Laser Modulation Signal is configured for DC operation (continuous wave), by setting the Q-Switch Period parameter to 0, then the ShootTime parameter is decoded as the length of time, in microseconds, to assert the Laser Modulation signal. On the other hand, if the Laser Modulation Signal is configured for pulse operation then the ShootTime parameter is decoded as the number of pulses to fire. The MD_shoot command must be preceded by a jump to establish the beam position; any other instruction will cause an error.

Record Layout:	<MD_shoot> <Shoot_Time>
Opcode:	0x00000018
Record Size:	8 Bytes
Parameters:	Shoot_Time
Restrictions:	Cannot be used within marking strokes.

Related Topics:
GCX file tutorial
Step and Shoot tutorial

MD_start_tracking

This command is used to start the Mark on the Fly counters. The MOF counters are finite in size and for a given MOF velocity, the counters will be valid for a finite time interval, so they must be activated and reset on a demand basis.

Record Layout:	< MD_start_tracking >
Opcode:	0x00000203
Record Size:	4 Bytes
Parameters:	None
Restrictions:	None

Related Topics:
GCX file tutorial
Mark on the Fly tutorial

MD_wait

This command will cause program operation to wait for the specified number of microseconds

Record Layout:	< MD_wait > <unit wait interval>
Opcode:	0x00000300
Record Size:	4+4 Bytes
Parameters:	Wait period in microseconds
Restrictions:	4,294,967,295 microseconds or about 50 days

Related Topics:
GCX file tutorial

MD_wait_begin_mark

This command will cause program operation to stop until the MARK_BEGIN signal is asserted

Record Layout:	< MD_wait_begin_mark >
Opcode:	0x00000301
Record Size:	4 bytes
Parameters:	None
Restrictions:	None

Related Topics:
GCX file tutorial

Parameter handling commands

WinMCL Plus allows the parameter sets or individual parameters to be adjusted within the gcx file. The following section addresses commands for handling parameters.

MD_parameter_set_begin

Marker to indicate the beginning of a parameter set.

Record Layout:	< MD_parameter_set_begin > <ulong parameter set number>
Opcode:	0x00000008
Record Size:	4 x 4 bytes
Parameters:	Parameter set number
Restrictions:	Parameter set number must be between0-63

Related Topics:
GCX file tutorial

MD_parameter_set_end

Marker to indicate the end of a parameter set.

Record Layout:	< MD_parameter_set_end >
Opcode:	0x00000009
Record Size:	4 bytes
Parameters:	None
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_step_period

This parameter sets the period of each microvector

Record Layout:	< MD_set_step_period ><long step_period>
Opcode:	0x00000041
Record Size:	4 + 4 bytes
Parameters:	Time period of each microvector in microseconds
Restrictions:	None.

Related Topics:
GCX file tutorial

MD_set_mark_size

This parameter sets the length of each mark microvector in LSB's

Record Layout:	< MD_ set_mark_size> <ulong mark size>
Opcode:	0x00000042
Record Size:	4 + 4 bytes
Parameters:	Mark size in LSB's
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_jump_size

This parameter sets the size in LSB's of each microvector jump.

Record Layout:	< MD_ set_jump_size> <ulong jump size>
Opcode:	0x00000043
Record Size:	4 + 4 bytes
Parameters:	Jump size
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_power

This parameter sets the 8 bit value sent via Port B to the laser to control output power.

Record Layout:	< MD_set_power ><long power>
Opcode:	0x00000044
Record Size:	4 + 4 bytes
Parameters:	Power in arbitrary units
Restrictions:	0-256

Related Topics:
GCX file tutorial

MD_set_intensity

This parameter determines the signal sent via Port N.

Record Layout:	< MD_ set_intensity><ulong intensity>
Opcode:	0x00000045
Record Size:	4 + 4 bytes
Parameters:	Intensity
Restrictions:	Value should be in the range 0-256

Related Topics:
GCX file tutorial

MD_set_polyline_delay

This parameter sets the delay between successive elements of a mark. This delay is inserted to allow marks to avoid rounding at corners.

Record Layout:	< MD_set_polyline_delay ><ulong polyline delay time>
Opcode:	0x00000046
Record Size:	4 + 4 bytes
Parameters:	Polyline delay time in microseconds
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_jump_delay

This parameter sets the delay in microseconds allowed for the system to settle after a jump

Record Layout:	< MD_ set_jump_delay><ulong delay.
Opcode:	0x00000047
Record Size:	4 + 4 bytes
Parameters:	Delay value in microseconds
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_stroke_delay

This parameter sets the period of time the galvos are held in position at the end of a stroke.

Record Layout:	< MD_ set_stroke_delay><stroke_delay>
Opcode:	0x00000048
Record Size:	4 +4 bytes
Parameters:	Stroke delay in microseconds
Restrictions:	Zero or positive value

Related Topics:
GCX file tutorial

MD_set_laser_on_delay

This parameter sets the period of time in microseconds between the galvo command that begins a mark and the laser modulation signal assertion. A negative value will assert the laser modulation signal before the command to move the galvo is sent.

Record Layout:	< MD_set_laser_on_delay ><long laser_on_delay>
Opcode:	0x00000049
Record Size:	4 + 4 bytes
Parameters:	Laser on delay in microseconds
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_laser_off_delay

This parameter sets the period of time between the time the laser modulation signal is unasserted and the end of the stroke delay interval.

Record Layout:	< MD_set_laser_off_delay ><long laser_off_delay>
Opcode:	0x0000004A
Record Size:	4 + 4 bytes
Parameters:	Laser off delay in microseconds
Restrictions:	Zero or positive value. Increasing the value in excess of the stroke delay interval has no effect

Related Topics:
GCX file tutorial

MD_set_power_delay

This parameter sets the waiting period in microseconds after a power change is commanded before marking continues. This is normally used to allow a flashlamp to stabilize.

Record Layout:	< MD_ set_power_delay><power_delay>
Opcode:	0x0000004D
Record Size:	4 +4 bytes
Parameters:	Power delay in microseconds
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_fps_enable

This flag enables or disables operation of the first pulse suppression trigger.

Record Layout:	< MD_set_fps_enable > <ulong flag>
Opcode:	0x00000060
Record Size:	4+4 bytes
Parameters:	Flag 0 for false non-zero for true
Restrictions:	None

Related Topics:
GCX file tutorial

MD_set_qswitch_period

This parameter sets the time allowed for the full period of a laser pulse. For example, a 10 kHz rate would be expressed as 100 microseconds.

Record Layout:	< MD_ set_qswitch_period>
Opcode:	0x00000061
Record Size:	4 + 4 bytes
Parameters:	Q-switch period in microseconds
Restrictions:	Setting the period to zero results in CW operation

Related Topics:
GCX file tutorial

MD_qswitch_width

This parameter sets the length of time in microseconds for each laser pulse. For example, a 50 percent duty cycle pulse in a 10 kHz system would be 50 microseconds long.

Record Layout:	< MD_ qswitch_width><ulong width>
Opcode:	0x00000062
Record Size:	4 + 4 bytes
Parameters:	Width of laser pulse in microseconds
Restrictions:	Width should be shorter than q-switch period. For CW, set q-switch period to zero.

Related Topics:
GCX file tutorial

MD_set_break_angle

This parameter sets the angle above which corners in marks will be broken into separate strokes.

Record Layout:	< MD_set_break_angle ><long break_angle>
Opcode:	0x00000064
Record Size:	4 + 4 bytes
Parameters:	Break angle in degrees
Restrictions:	None

Related Topics: GCX file tutorial. Break Angle

MD_set_dither_width

This parameter sets the width of a dither line

Record Layout:	< MD_set_dither_width ><long dither_width>
Opcode:	0x00000065
Record Size:	4 + 4 bytes
Parameters:	Dither Width in LSB
Restrictions:	None

Related Topics: GCX file tutorial, Dither Width

MD_set_dither_feed

This parameter sets the feed rate for a dither pattern.

Record Layout:	< MD_set_dither_feed ><long dither_feed>
Opcode:	0x00000066
Record Size:	4 + 4 bytes
Parameters:	Dither feed in LSB
Restrictions:	None

Related Topics: GCX file tutorial, Dither Feed

GLOSSARY

The glossary provides definitions of commonly used terms.

Absolute Coordinate

An absolute coordinate is defined as the distance from the origin of the coordinate system to the point.

Beam Path

The beam path is the continuous motion of the focal point along the plane of projection.

Beam Positioning Lag

Beam Positioning Lag comes about because the host computer is not in a tight position feedback loop with the mirror position. Typically, the computer issues commands to the servo, via the HC/3, which correspond the desired destination. The servos then attempt to aim the mirrors in the appropriate way and as fast as possible but they never operate at the same rate as the host computer due to the mechanical consideration of mirror inertia or the electrical consideration of servo bandwidth and transmission latency.

BEGIN_MARK signal

The BEGIN_MARK signal is an input to the HC/3. This signal can be used to allow a hardware input to commence a mark job. The signal is active low. The signal input is located at pin 4 of the optically isolated connector I2, or pin 17 of the internal connector I3. The following diagram shows the location of the BEGIN_MARK signal on both connectors.

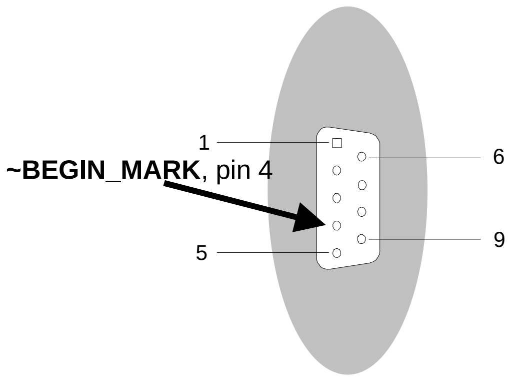


Figure 34: Location of the Begin_Mark signal on the pin out of I2.

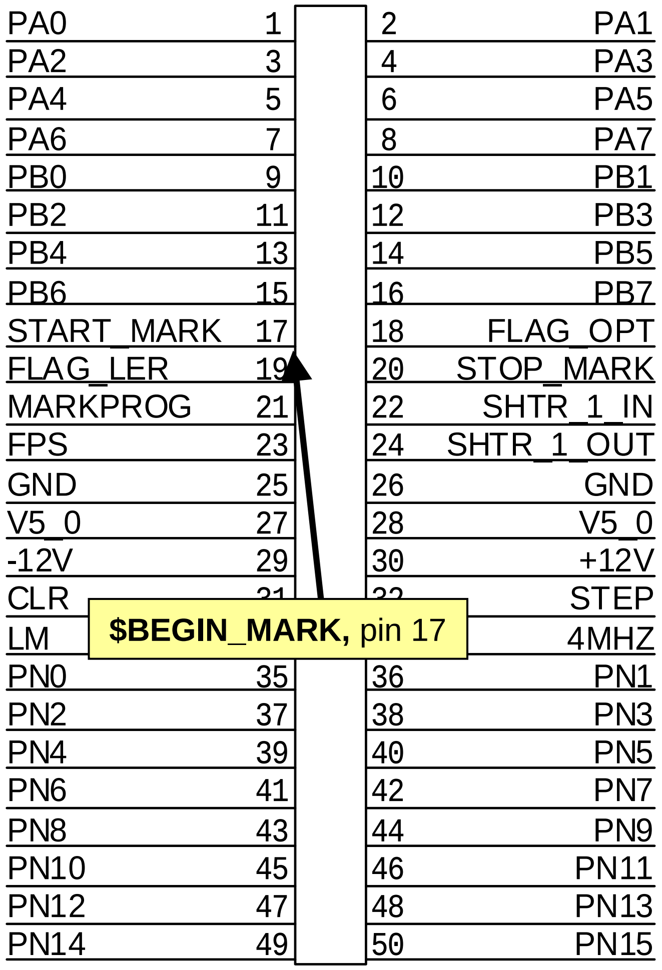


Figure 35: Location of the Begin_Mark signal on the pin out of I3.

There is no MD command related to BEGIN_MARK. The MC commands related to the BEGIN_MARK signal are: MC_get_begin_mark and MC_start_mark_on_begin.

Related topics:

- MC_get_begin_mark
- MC_start_mark_on_begin
- Hardware IO

Exchange Flag

The Exchange Flag is set when the X and Y-axis's are exchanged. This results during the execution of the MC_set_xy_exchange function. The state of the Exchange Flag can be read back with the MC_get_xy_exchange function. The Exchange flag is a global parameter; it is not affected when a new parameter set is selected.

FPS Delay

FPS Delay is used during laser control when the FPS Enable flag is set true. The value of the parameter defines a delay in the FPS pulse when the laser is turned ON. FPS Delay is one of the global parameters; it is not affected when a new parameter set is selected.

Related topics:
Laser Power Control
FPS Enable

Field Distortion Correction Table

The Field Distortion correction table is supplied by General Scanning for a given head design and lens option. The table consists of a row of numbers that define interpolations that 'square up' the marking field when it is projected onto the plane of projection. When stored in the proper location in WinMCL, the Field Distortion Correction Table is a member of the Global Parameters; it is not affected when a new parameter set is selected.

Related Topics:
Field Distortion Correction Tutorial
MC_set_corrtable
MC_set_corrtableFromMemory
MC_get_corrtable

Flip Matrix

The Flip Matrix is used to flip either the X or Y axis or both. The Flip Matrix is a member of the Global Parameters; it is not affected when a new parameter set is selected.

Focal Point

The focal point is the focus of the laser beam directed by the angular displacement of the steering mirrors.

GCX File and Data

GCX files contain the marking instructions required to perform a marking job. GCX files simply store the sequence of marking instructions required to perform a piece of a job. GCX files can call other GCX files, and they can be loaded and executed out of memory.

GCX Data is a buffer with list of GCX commands. The application is responsible of loading the GCX Data into the memory and then passing the pointer to GCX Data to WinMCL Plus. The GCX Data in memory should stay valid until WinMCL Plus processes it.

Global Matrix

The Global Matrix is represented as 3x3 matrix of floating point numbers:

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 36: Locations of the parameters for the Global Matrix

The MD_global_matrix GCX command is an input method where the parameters of the instruction are directly mapped to the matrix locations as shown in the diagram above. The MD_global_transform GCX provides an alternate input method for the parameters. The following diagram illustrates the procedure invoked on the parameters of this function.

$$\begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 37: Locations of the parameters for the alternate input method.

The Global Matrix is one of the Global Parameters; it is not affected when a new parameter set is selected.

Global Parameters

The Global Parameters remain unchanged when the parameter set is changed. Most of the Global Parameters are in fact the transformation matrices. Also, each HC/3 card has its own Grid Correction Table and Power Table and these are treated as Global Parameters also. The list of Global Parameters is:

- FPS Delay
- Universal Matrix
- Global Matrix
- Part Matrix
- Object Matrix
- Flip Matrix
- Exchange Flag
- (4) Grid Correction Tables
- (4) Power Tables

HC/3 device driver

The HC/3 device driver is typically installed when the HC/3 card is installed in the computer. The WinMCL Plus software communicates with the HC/3 driver to transfer data from GCX file to the output ports of the HC/3 card.

HC/3

The HC/3 is a PCI bus card that is used to command the marking head. WinMCL Plus communicates to the HC/3 device driver through function calls and the device driver communicates to the HC/3 card via the job queue. Please refer to General Scanning manual # 7OM-034 for complete information on the HC/3 card. At the hardware level the HC/3 card communicated via a serial link to either of the General Scanning HPM or HCI cards translator cards. The HC/3 card also has a DB9 connector for certain inputs and output that can be read and written from WinMCL Plus.

Related topics:

- BEGIN_MARK signal
- MARK_ABORT signal
- REMOTE_EXECUTE signal
- MARK_ERROR signal
- MARK_IN_PROGRESS

Job Queue

There are two ways to process more than one job.

- The first approach is to use MC_load_job / MC_load_job_from_memory and MC_start_mark for each job.

  ```
  Do until last job
  {
          MC_load_job_from_memory
          MC_start_mark
  }
  ```

- The second approach is to use the Job Queue. The application loads a number of jobs by MC_load_job / MC_load_job_from commands and then issues MC_start_mark command. While WinMCL processes the loaded jobs the application may load more jobs into the Job Queue. The application should use two threads, because MC_start_mark doesn't return until marking is done. The example of two-thread pseudo code is represented below.

```
        Thread 1                                Thread 2
```

```
For (i = 0; I < M ; I++)                          MC_start_mark
{                                             n = MC_get_job_count
        Create GCX file
        MC_load_job_from_memory
        If (i == m)
           Activate Thread 2                  if (n > 0)
}                                                     MC_start_mark
```

The first thread creates m GCX files, loads them and activates thread 2. Then it keeps calculating rest M – m files and loads them. The second thread starts the marking process. When MC_start_mark returns, the second thread checks the number of jobs is the Job Queue. If the n - number of jobs in the queue is equal zero, then all the jobs were processed. If the n is more than zero, then last n jobs weren’t processed. To process the rest of the jobs in the job queue the application should issue another MC_start_mark. If application doesn’t want to process rest of the job queue, it issues MC_reset_job_queue command.

Jump Delay

The jump delay occurs at the end of a jump motion. The delay is programmed in the parameter set by adjusting the jump delay parameter. The jump delay is useful in the coordination of the laser ON command with the start of a mark vector. Typical marking artifacts that can be removed by adjusting the jump delay include 'hooks' and 'blooms' at the start of strokes.

Jump Rate

The jump rate is the velocity of the focal point during jump motions. It is defined by two parameters: jump size divided by step period.

Jump Vector

A jump vector is a type of macro vector that moves the beam position while the laser is OFF. The velocity of the focal point during the motion is given by the jump rate.

The MCL commands that cause jumps to execute are:

- MC_jump_rel
- MC_jump_abs

The GCX MD commands that cause jumps to execute are:

- MD_jump_rel
- MD_jump_abs

Laser Modulation

Laser modulation refers to a type of control mechanism that allows you to control the power of a laser with an electrical signal, voltage, or parallel binary data.

Laser Modulation Signal

The Laser Modulation Signal is output on pin 2 of connector I2 as an optically isolated signal. The Laser Modulation Signal is also output on pin 33 of connector I3 in non-isolated form. In the WinMCL/HC/3 system, modulation refers to the ability to control laser power with a pulse waveform, the Q-Switch or Laser Modulation signal. Since this is a type of Pulse Width Modulation we call the signal the Laser Modulation Signal. It is important to remember that during strokes, the Laser Modulation signal configuration remains constant; it is not possible to change it while a mark vector is executing.

If the Q-Switch Period is set to 0, the Laser Modulation Signal will be a DC waveform.

Related topics:
Hardware IO
Laser Power Control
Q-Switch Period
Q-Switch Width

Laser Power Calibration Table

The Laser Power Calibration table is most often used in installations that contain more than one HC/3 and consequently more than one laser, possibly different makes and hours in service. The table is used to correct for differences in power output between lasers such that each marking head marks the part identically. Each HC/3 in a multiple card installation can have its own Laser Power Calibration Table. Single HC/3 systems must still use a Laser Calibration Table, but typically the default table is used.

The Laser Power Calibration Table is a lookup table addressed by the Power parameter in the Parameter Set. The value indexed is sent to the Port B output as a byte value and this is used to control the power of an external laser.

Related topics:

- Laser Power Calibration Table default value
- Hardware IO
- Port B output
- Master HC/3
- Data Structure Definitions

Laser Power Calibration Table default value

After MC_init or MC_reset the Laser Power Table takes on the default value, as shown in the following table.

Index	Value
0	0
1	1
2	2
3	3
4	4
etc...	etc...

The default power table simply provides a 1 to 1 correction factor for laser power, i.e. the index value equals the indexed value.

Laser Turn-ON Lag

Marking lasers typically have delays associated with laser ON and Laser OFF events. In typical operation, the HC/3 sends a command to the laser through the laser modulation signal. The laser must then transition from the OFF mode to the ON mode (start to lase) and this may take time.

Macro Vector

In the WinMCL Plus system, a macro vector is defined as the vector motion that is commanded by the MD mark, move and jump commands. Marking routines are conceived and programmed using macro vectors. During operation the macro vectors are converted to micro vectors at the hardware level and this determines the beam path velocity. The following picture shows the relationship between macro vectors and micro vectors.

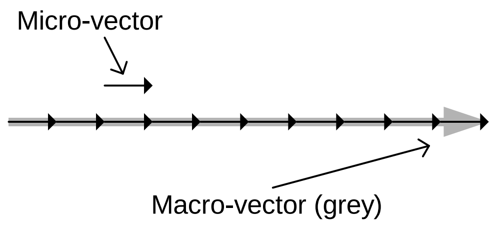


Figure 38: Graphic illustrates the composition of a macro vector.

MARK_ABORT signal

The MARK_ABORT signal is an input to the HC/3. This signal can be used to allow a hardware input to abort marking. The signal input is located at pin 3 of the External I/O connector. The signal is active low. The following diagrams show the location of the MARK_ABORT signal on the optically isolated connector I2 and the internal connector I3.

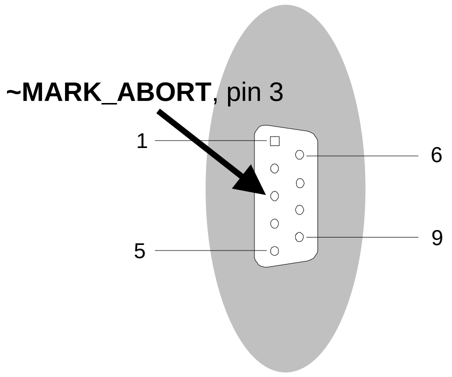


Figure 39: Location of the Mark_Abort signal on the pinout of I2.

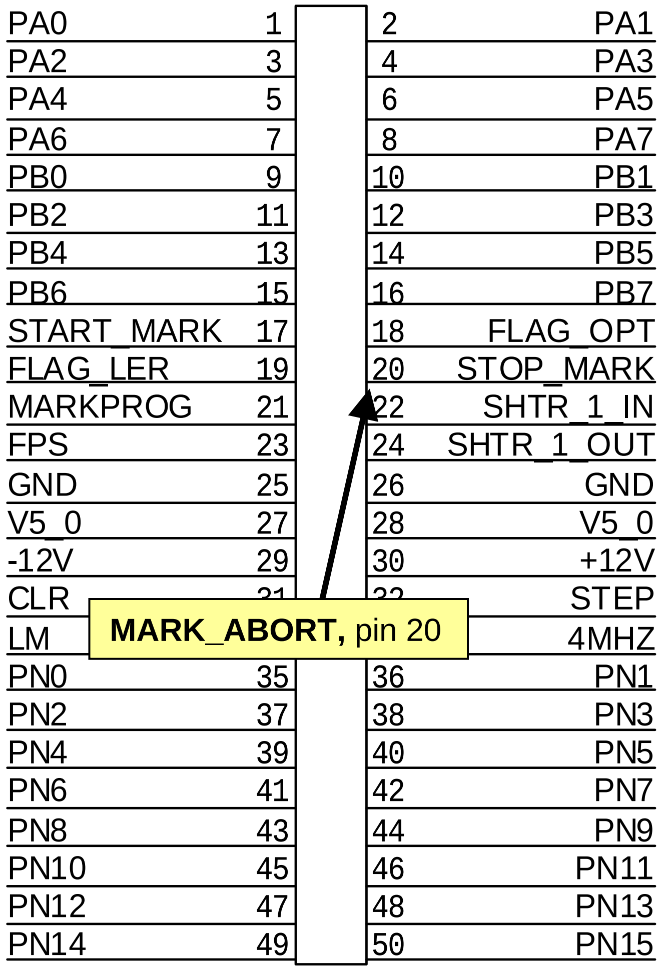


Figure 40: Location of the Mark_Abort signal on the pin out of I3.

There is no MD command related to MARK_ABORT. The MC command that reads MARK_ABORT is:
MC_get_mark_abort

Related topics:

- MC_get_mark_abort
- Hardware IO

Mark on the Fly

Mark on the Fly is an operational mode of WinMCL that, when enabled, causes the Marking Field to move along the plane of projection at a constant velocity. This feature is used to mark parts that are moving on an assembly line.

Mark Rate

In vector mode the mark rate is the velocity of the focal point during mark and move motions. It is defined by two parameters, mark size divided by step period. Since the step period is a common parameter between the mark and jump rates, it is only possible to differentiate the two with the mark size and jump size parameters. See the section on vector marking for more information.

In raster mode the mark rate is determined by the Q-Switch period, the number of pixels in the line and the length of the line. See the section on raster marking for more information.

Mark Vector

A mark vector is a type of macro vector that moves the beam position while the laser is ON. The velocity of the focal point during the motion is given by the mark rate. Strokes are made up of mark vectors.

All marking must be done with GCX files; there are no MCL commands that will cause a mark vector to execute.

The GCX MD commands that cause a marking vector to execute are:

- MD_mark_rel
- MD_mark_abs

MARK_ERROR signal

The MARK_ERROR signal is an output from the HC/3. This signal can be used to signal external hardware that WinMCL Plus has detected an error. The signal is cleared by the MC_reset function and set by either MD_set_mark_error or MC_set_mark_error.

The signal output is located at pin 9 of the External I/O connector. The signal is active low. The following diagram shows the location of the MARK_ERROR signal on the optically isolated connector I2.

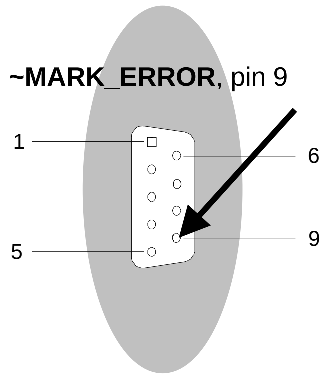


Figure 41: Location of the Mark_Error signal on the pin out of I2.

The MD command related to MARK_ERROR is:

- MD_set_mark_error

The MC command related to MARK_ERROR is:

- MC_set_mark_error

Related topics:

- MD_set_mark_error
- MC_set_mark_error
- Hardware IO
- MC_reset

MARK_IN_PROGRESS

The MARK_IN_PROGRESS signal is an output from the HC/3. This signal is used by WinMCL Plus to tell external hardware that a marking job is in progress. The signal is active low and is set when marking a job. The MARK_IN_PROGRESS signal is reset when either the MC_abort_mark or MC_reset commands are executed or if the MARK_ABORT signal is asserted.

The signal output is located at pin 5 of the optically isolated connector I2, or pin 21 of the internal connector I3. The following diagram shows the location of the MARK_IN_PROGRESS signal on both of the connectors.

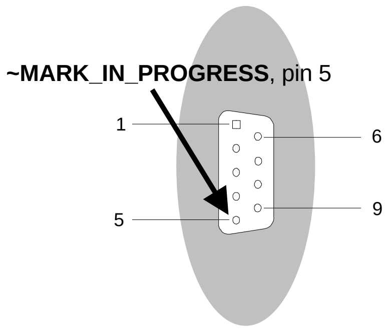


Figure 42: Location of the Mark_in_Progress signal on the pin out of I2.

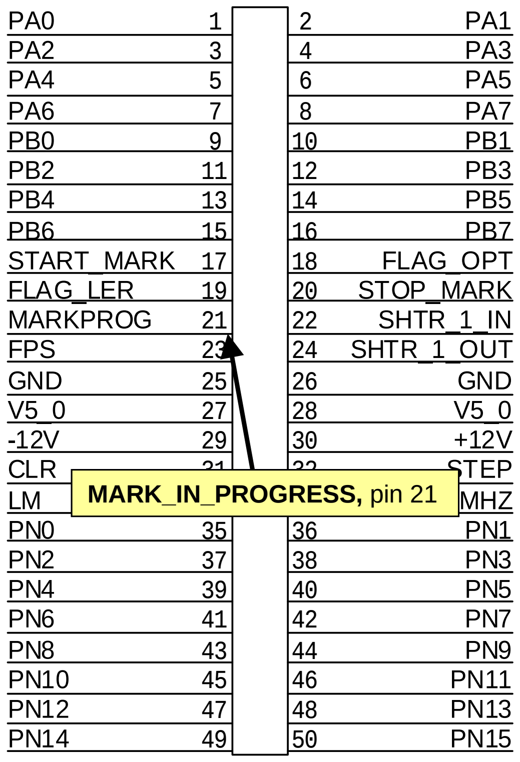


Figure 43: Location of the Mark_In_Progress signal on the pin out of I3.

Related topics:

- Hardware IO
- MC_abort_mark
- MC_reset
- MARK_ABORT signal

Marking Field

The marking field is the area on the plane of projection that the marker head can mark. The size of the marking field is limited by the distance of the marking head to the plane of projection, the angular displacement of the mirrors in the head and any effects generated by the lens. Uncorrected marking field are shaped like mirrored hyberboli, somewhat like the hourglass shape. Correction of this distortion can be done with lenses or in software with the Field Distortion Correction Table.

Marking Job

A marking job consists of a number configuration instructions in WinMCL and motion instructions stored in the GCX format either in memory or in a file. A marking job traditionally means all of the operations required to perform a complete mark on the part.

Master HC/3

In multiple card installations the Master HC/3 serves as the controller for up to three slave cards. The slaves are controlled via the I/O signals. The master HC/3 in a multi-card installation is identified with a board ID of 0.

Micro Vector

Micro vectors are generated in the HC/3 as part of interpreting the instruction stream of macro vectors during a marking job. Two types of micro vectors can be defined, those that are used in jump vectors and those that are used in mark vectors; thus there are essentially two types of micro vectors, distinguished by the velocity of the focal point during execution.

Micro vectors are defined with distance and time parameters and the result, dist/time = velocity, is the beam velocity. For jumps the parameters of interest are the jump size and the step period. For marks the parameters of interest are the mark size and the step period. Please note that the step period parameter is a common factor to the definitions of both the jump micro vector and the mark micro vector so it is useful to consider it as a speed scaling parameter; when the value is made smaller the speed will increase. The jump size and mark size parameters determine the ratio of the jump rate to the mark rate. Typically, the jump rate is much faster than the mark rate because the laser is OFF and we are not concerned about marking artifacts.

Mirror Inertia

Mirror inertia describes how difficult it is to turn the mirror on the axis of the galvo and it is based on the physical characteristics of the mirror including the mass and dimensions. Mirror inertia is not directly addressed in WinMCL as it is part of the physical plant: the servo, galvo and mirror. Additional information regarding mirror inertia can be found on the General Scanning web site or by inquiries through your sales representative.

Move Vector

A move vector is a type of macro vector that runs at the same velocity as the mark vector, i.e. the mark rate, but the laser is OFF during the motion. This type of motion is useful in establishing a constant beam velocity before turning the laser ON.

All move vectors are commanded from GCX files; there are no MCL commands that would cause a move vector to execute.

The GCX MD commands that cause a move vector to execute are:

- MD_move_rel
- MD_move_abs

OPT2 signal

The OPT2 signal is an input from the HC/3. This signal input has been traditionally used for laser error input for those lasers that have to signal errors in hardware. The signal input is located at pin 7 of the External I/O connector. The signal is active low. The following diagram shows the location of the OPT2 signal on the connector.

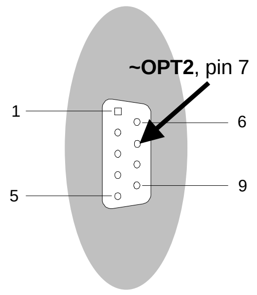


Figure 44: Location of the OPT2 signal on the pin out of I2

The MC command related to this port is:
MC_get_optional_status

Related topics:

- MC_get_optional_status
- Hardware IO

Object Matrix

The Object Matrix is traditionally used to transform graphic objects within a job. For instance, the job requirement might be that a serial number be written on one edge of a square, model number on another edge and the corporate logo on a third edge. See the diagram.

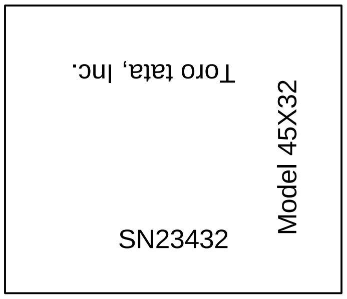


Figure 45: Example of part marking, with three graphic objects in different orientations.

Changes in the Object Matrix, that precede the marking instructions for certain strings, allow the string routines to be written in the normal, upright sense. The Object Matrix simple rotates the coordinate system as needed to mark the mini job in a given orientation. The Object Matrix is represented as 3x3 matrix of floating point numbers as shown in the following diagram:

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 46: Location of the parameters in the Object Matrix

The MD_object_matrix GCX command is an input method where the parameters of the instruction are directly mapped to the matrix locations as shown in the diagram above. The MD_object_transform GCX provides an alternate input method for the parameters. The following diagram illustrates the procedure invoked on the parameters of this function.

$$\begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 47: Location of the parameters in the matrix for the alternate input method.

The Object Matrix is one of the Global Parameters; it is not affected when a new parameter set is selected.

Related topics:

Marking Field Transformations
MD_object_transform
MD_object_matrix

Pacing Delay

Pacing delays are inserted into the GCX marking instruction stream to control the rate of execution of marking commands in the computer in relation to the electrical and mechanical capabilities of the marking head and laser. When you are assembling a GCX file for marking you do not need to explicitly insert the pacing delays, they are side effect of certain commands or combinations of commands. The delay time is typically controlled by a parameter in the parameter set. The following parameters control various pacing delays:

- Jump Delay
- Laser Off Delay
- Laser On Delay
- Poly Line Delay
- Power Delay
- Stroke Delay

Parameter Set

Parameter sets provide a convenient way to modify a number of distinct parameters all at once, and also to change just one or two at a time. In previous versions of WinMCL Plus, the parameter set was called the Laser Parameter Set, because all of the parameters were directly related to control of the laser. However, this version of WinMCL Plus includes parameters that control the mark on the fly feature, so the employment of parameters has been expanded beyond control of the laser. Parameter sets are no longer supported as data in GCX Files.

Related topics:

- Parameter Set Tutorial
- Parameter Set default values
- List of Parameters

Parameter Set default values

The default parameter set has the following values:

Parameter	Default Value
BreakAngle	180
DitherFeed	0
DitherWidth	0
FPSEnable	1
Intensity	0
JumpDelay	1000
JumpSize	50
LaserOffDelay	100
LaserOnDelay	100
MarkSize	50
PolyLineDelay	80
Power	0
PowerDelay	1500
QSwitchPeriod	100
QSwitchWidth	75
ShootTime	100
StepPeriod	20
StrokeDelay	1000
TargetXVelocity	0
TargetYVelocity	0

Part Matrix

The Part Matrix is traditionally used align the Marking Field to the physical part to be marked. Depending upon the type of material handler that is placing the part under the marker the Part Matrix may be either static or dynamic from part to part, but during the marking job for one part the Part Matrix does not change value. The Part Matrix is represented as 3x3 matrix of floating point numbers:

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 48: Locations of parameters in the Part Matrix

The MD_part_matrix GCX command is an input method where the parameters of the instruction are directly mapped to the matrix locations as shown in the diagram above. The GCX command MD_part_transform provides an alternate input method for the parameters. The following diagram illustrates the procedure invoked on the parameters of this function.

$$\begin{bmatrix} \cos(ang) & -\sin(ang) & 0 \\ \sin(ang) & \cos(ang) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 49: Location of parameters in the matrix for the alternate input form.

The Part Matrix is one of the Global Parameters; it is not affected when a new parameter set is selected.

Related topics:
Marking Field Transformations
MD_part_matrix
MD_part_transform

Plane of Projection

The plane of projection extends infinitely in all directions from the marking field. In the most useful geometries, the beam that is projected from the marking head when the mirrors are aimed at X = 0, Y = 0 is normal to the plane of projection.

Poly Line Delay

A polyline is composed of more than one mark or vectors. The polyline delay is applied to the end of each macro vector (vertices at which the beam path moves in a new direction) except for the last macro vector in a stroke. The polyline delay is useful in establishing sharply marked corners during the changes in direction of a stroke. The following figure shows where the polyline delay occurs inside a stroke.

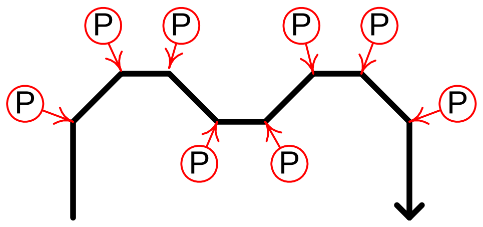


Figure 50: A representation of a stroke with the locations of polyline delays indicated.

See also:

- Stroke
- Stroke Delay

Port A input

The PORT A input register is located on the internal I/O connector of the HC/3. Use this to input byte values from an external source into WinMCL Plus.

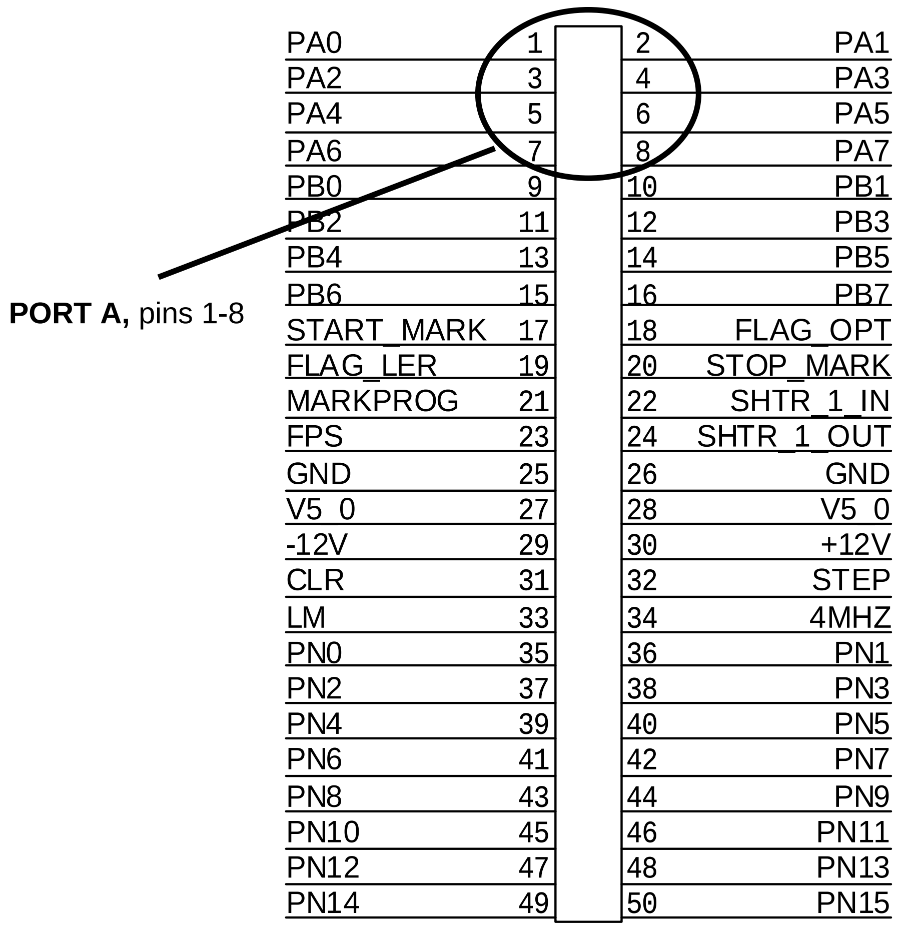


Figure 51: Location of Port A on the pin out of I3.

There is no MD command related to the Port A input, typically because there are no provisions in GCX files for handling input data.

The MC command for reading Port A is MC_get_portA_input.

Related topics:

- MC_get_portA_input
- Hardware IO

Port B output

Port B is used to control laser power. It is an 8-bit output port located on the internal I/O connector of the HC/3 as shown in the following diagram.

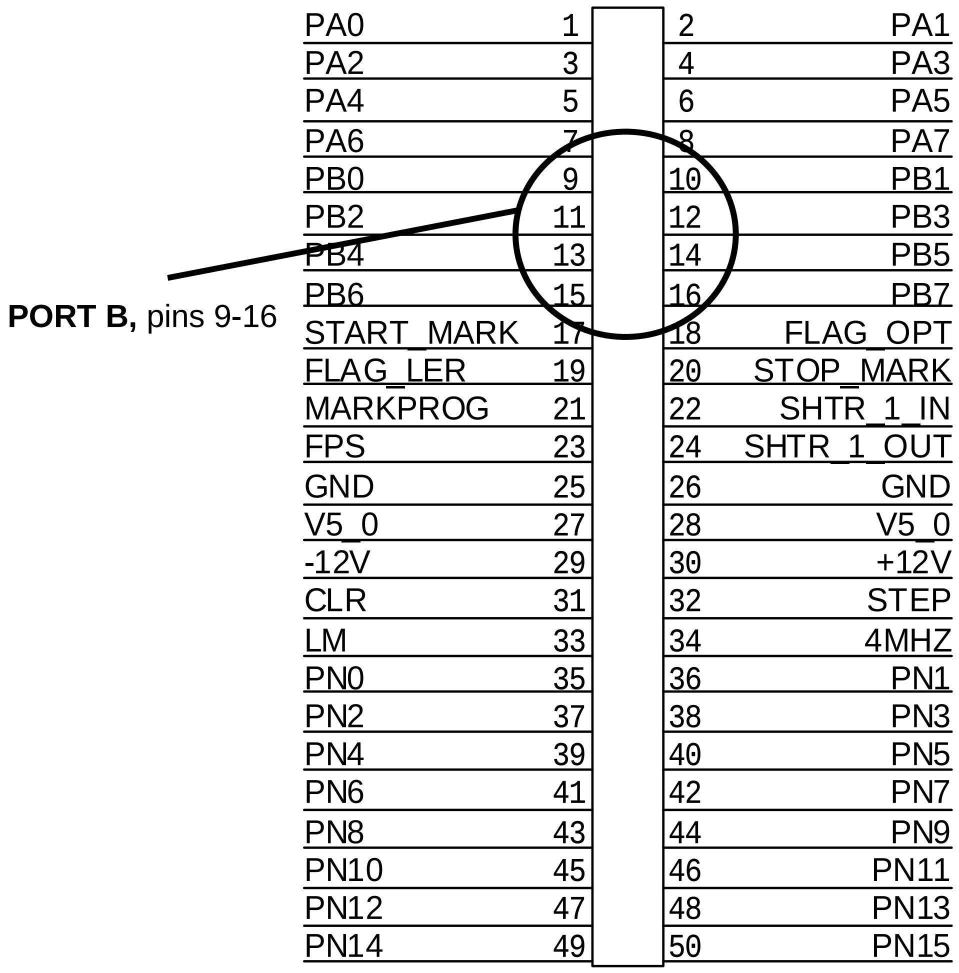


Figure 52: Location of Port B on the pin out of I3.

The optional General Scanning Port adapter card, Part #272.216.00, can be used to bring the Port B outputs out the computer expansion bus. The pin out for this connector is shown in the following diagram.

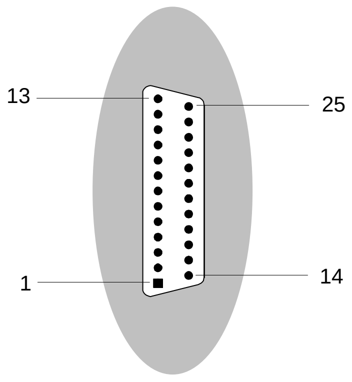
INTERFACE	PIN	ASSIGNMENT
 25 Pin D-Sub male connector	1	PORT BO
	2	PORT B1
	3	PORT B2
	4	PORT B3
	5	PORT B4
	6	PORT B5
	7	PORT B6
	8	MSB (/LP_COUT)
	9	NC
	10	FPS TRIGGER
	11	SHUTTER IN
	12	GROUND
	13	+5 V
	14	GROUND
	15	NC
	16	+5V
	17	NC
	18	NC
	19	NC
	20	NC
	21	GROUND
	22	+5 V
	23	SHUTTER OUT
	24	LASER MODULATION
	25	NC

Figure 53: Location of Port B on accessory bulkhead connector.

The laser power data output by this port is controlled in software via the Power parameter of the parameter set. When the value of the Power parameter is changed in software, the registering of data to Port B is immediate, but Power Delay delays further execution of FIFO instruction. The Power Delay pause interval is a pacing parameter that allows the WinMCL software to stay in step with the physical and electrical characteristics of the marking laser during a change in power. Proper utilization of the Power Delay parameter insures that subsequent marking instructions will execute with constant power.

Related topics:

- MC_reset_powertable
- MC_set_powertable
- Power
- Power Delay
- Hardware IO

Port N output

Port N is used to control laser power or other general purpose digital circuitry. The port is 16 bits wide and it is located on the internal I/O connector of the HC/3 as shown in the following diagram.

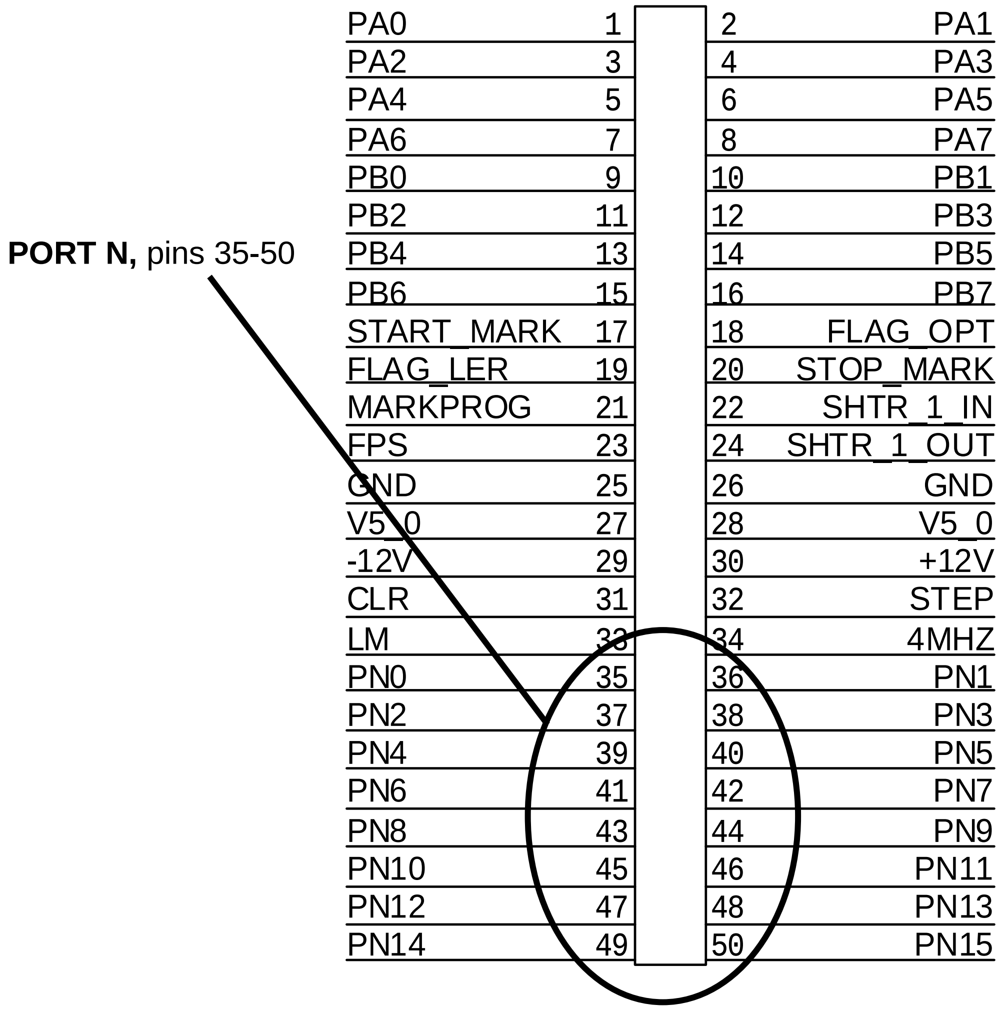


Figure 54: Location of Port N on the pinout of I3.

See also:

- Hardware IO

Raster Mode

Raster mode is used for marking a line of dots on the part. The timings required by the raster marking commands involve changes to the Q-switch period and laser delay parameters so it is considered a separate mode from vector. Also, the velocity of the focal point is not set by the mark size and Step Period, as done in vector mode operations, so it is important that all parameters be carefully coordinated when switching modes. Please see the section on raster mode operations in the tutorial.

Raster Pulses

Raster pulses are laser control pulses that are output from the LM signal or PortB, or both. These pulses are output only during the execution of the MD_raster_abs and MD_raster_rel commands. Configuration of the pulse output is performed with the MC_set_raster_mode command prior to the execution of the marking job. See the tutorial on raster marking.

Relative Coordinate

A relative coordinate is defined as the distance from the current position.

REMOTE_EXECUTE signal

The REMOTE_EXECUTE signal is an output from the HC/3. This signal can be used to control external hardware under the command of WinMCL Plus. The signal output is located at pin 8 of the External I/O connector. The signal is active low. The following diagram shows the location of the REMOTE_EXECUTE signal on the optically isolated connector I2.

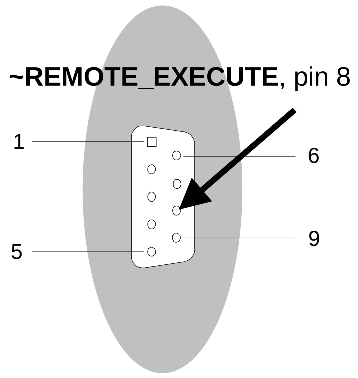


Figure 55: Location of the Remote_Execute signal on the pin out of I2.

The MD command related to REMOTE_EXECUTE is: MD_set_remote_execute

The MC command related to REMOTE_EXECUTE is: MC_set_remote_execute

See also:

- MD_set_remote_execute
- MC_set_remote_execute
- Hardware IO

Servo Bandwidth

Servo bandwidth is a number that describes how fast and accurate the galvos can move the mirrors. The higher the bandwidth number, the fast the servo, and the more characters per second can be marked. Servo bandwidth is not directly related to WinMCL; rather, it is a characteristic of the "plant"; i.e. the mirror, galvo and servo. For more information on servo bandwidth please visit the General Scanning web site or talk to your service representative.

Step Period

The step period determines the rate at which instruction events are read and written from the host computer. More importantly, the step period is a factor in the mark rate and jump rate during marking operations. Ultimately, the step period is the system 'heartbeat'.

Related topics:

- Step Period
- Jump Rate
- Mark Rate

Stroke

A stroke is composed of any number of consecutive mark vectors. In a sense, a stroke is considered to be a poly line.

Stroke Delay

The stroke delay is applied at the end of a stroke. This delay takes into account the laser OFF time as it allows the laser power to diminish to a non-marking level. The stroke delay is controlled via the parameter set by adjusting the value of the stroke delay parameter. Typical marking artifacts that can be removed with the stroke delay include 'tail' or 'trailing lines' at the end of strokes.

Related topics:

- Stroke
- Stroke Delay

Power Table

The Power Table is a 256 entry lookup table that is addressed by the value of the Power parameter. The value of the table index by the Power parameter is sent to Port B to control the laser power. If the power table is changed and the value indexed by the Power parameter is not the same as in the previous table then the Power Delay is executed. The Power Table is a member of the Global Parameters; it is not affected when a new parameter set is selected.

Power Table default value

The default Power Table has an equivalent correspondence between the index of the table and the members value. For example pt[0] = 0, pt[10] = 10, etc. In this form, the power table offers no scaling or correction; the value of the Power parameter is the value sent out Port B.

Transmission Latency

The HC/3 card commands the galvos through the use of a daughter card that resides inside the marking head. The two cards communicate with each other with a serial protocol; consequently, there is a delay between the time of execution of an instruction in the job queue and the time of generation of the command voltage inside the marking head. This delay is particularly important in operations that involve the coordination of signals between the HC/3 card (typically the Laser Modulation signal) and the position command voltages generated in the daughter card.

Universal Matrix

The Universal Matrix is traditionally used to establish the fundamental geometry of the marking heads to the marking field. As such, the GCX files need not concern themselves about the particulars of the installation, but rather assume that a coordinate system has been established, so there are no MD commands that manipulate the Universal Matrix. The following diagram shows the row and column locations for each of the input parameters for the Universal Matrix.

$$\begin{bmatrix} A & B & X0 \\ C & D & Y0 \\ 0 & 0 & 1 \end{bmatrix}$$

Figure 56: Matrix format of input parameters of Universal Matrix

The WinMCL function MC_set_xform_matrix is an input method where the parameters of the instruction are directly mapped to the matrix locations as shown in the diagram above. If you want to read back the values stored in the matrix use the MC_get_xform_matrix.

```
MC_set_xform_matrix( a, b, c, d, x0, y0 );
```

Traditionally, the Universal Matrix is programmed by the host program, perhaps from values stored in an initialization files, or perhaps by values determined through a calibration process. The Universal Matrix is one of the Global Parameters; it is not affected when a new parameter set is selected.

Related topics:
Marking Field Transformations
MC_set_xform_matrix
MC_get_xform_matrix

Vector

Typically, vectors are treated as free vectors. This means that the tail of the vector does not have to be fixed at the origin. Beam paths are composed of sequences of free vectors, or vectors, where the tail of the next vector is coincident with the head of the previous.

Vector Mode

The vector mode of operation involves marking lines on the object by moving the mirrors while the laser is ON. In typical operation the beam is steered in the X and Y axis and the laser is focused in the Z-axis. During the marking motion the laser power remains constant.

WinMCL Plus Default State

See:

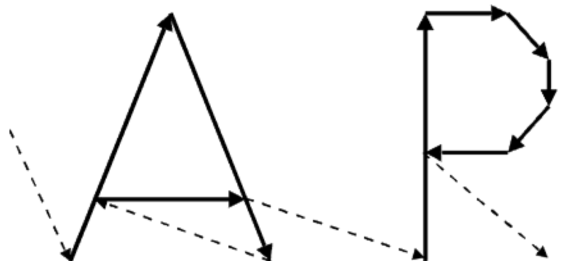
- Parameter Set default values
- Laser Power Calibration Table default value

APPENDIX A: ADJUSTING PARAMETERS

Galvanometer and laser control programs include a variety of laser parameters requiring fine adjustment for optimal system performance. The following section describes laser parameters, their application effects, and tips on adjusting each parameter.

Description of Laser Parameters

Scanning Vectors



Marking Vectors (Stroke)

- Laser beam is **on**
- Marking speed is determined by the material to be marked
- Speed is set by step size and step period

Non-Marking Vectors (Jumps)

- Laser beam is **off**
- Positioning speed to move from one set of marking vectors to the next
- Speed is set by jump size and step period
- Generally faster than Marking Vectors to improve throughout
- Consecutive jumps are combined at run time

Scanning Velocity

Galvo and Beam Velocity

$$\text{Velocity} = \frac{\text{Distance}}{\text{Time}}$$

- How fast does the galvo move?

Distance = step size = # of LSBs per DAC command [LSBs]
Time = step period= time for one DAC command [µsecs]

$$\text{Galvo Velocity} = \frac{\text{Distance}}{\text{Time}} = \frac{\text{step size}}{\text{step period}}$$

- How fast does the beam move?

  field size = 65535 LSBs

$$\text{Distance in the field} = \frac{\text{field size}}{65535} \times \text{stepsize}$$

$$\text{Beam Velocity} = \frac{\text{Distance in the field}}{\text{Time}} = \frac{\text{step size}}{\text{step period}} \times \frac{\text{field size}}{65535}$$

Calculating Marking Speed (Beam Velocity)

$$\frac{\text{step size}}{\text{step period}} \text{ x } \frac{\text{field size}}{65535} = \text{marking speed}$$

$$\text{eg. } \frac{52 \text{ lsbs}}{100\ \mu\text{sec}} \text{ x } \frac{108 \text{ mm}}{65535} \text{ x } \frac{1.000.000\ \mu\text{sec}}{1 \text{ sec}} = 857 \text{ mm/sec}$$

Calculating Jump Speed

$$\frac{\text{jump size}}{\text{step period}} \text{ x } \frac{\text{field size}}{65535} = \text{jump speed}$$

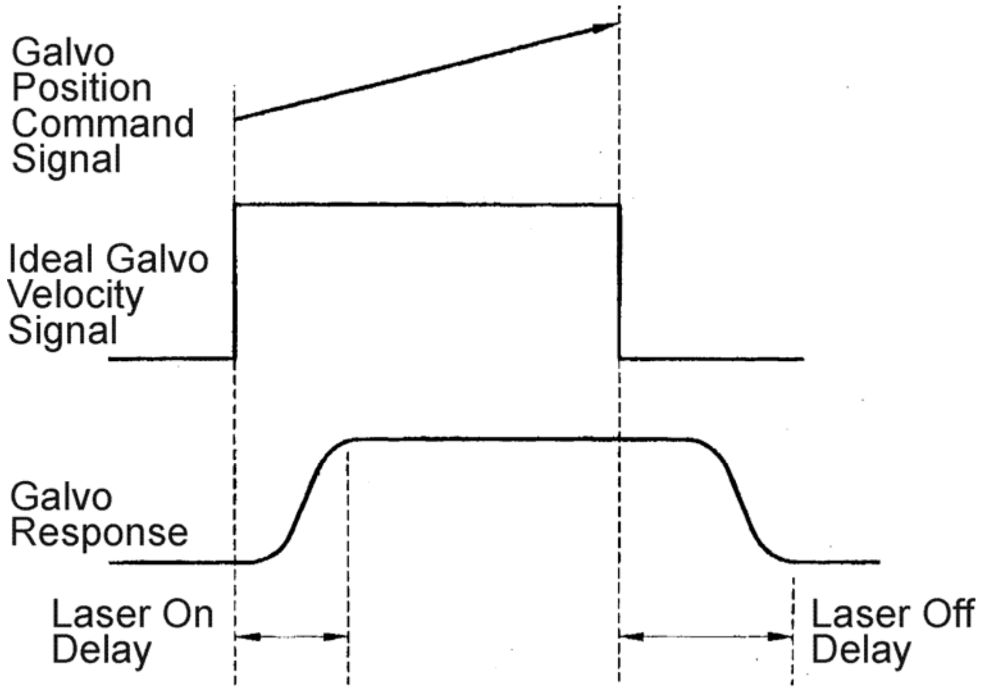
$$\text{eg. } \frac{150 \text{ lsbs}}{100\ \mu\text{sec}} \text{ x } \frac{108 \text{ mm}}{65535} \text{ x } \frac{1.000.000\ \mu\text{sec}}{1 \text{ sec}} = 2472 \text{ mm/sec}$$

Delays

Laser On/Off Delay

- Inertial mass of galvos and mirrors causes a time lag in response to command signals
- Laser on and laser off delays are time delays to allow galvos to respond to command signals
- Synchronizes switching the laser beam on and off with galvo movement
- Settings dependent upon scan speed

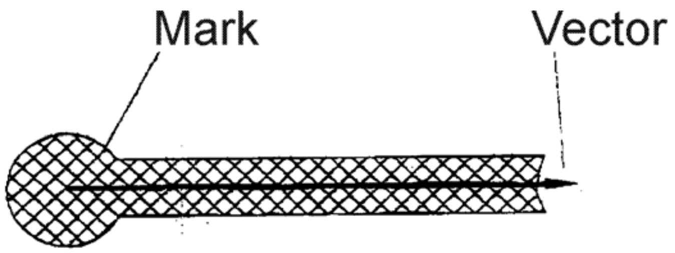
<u>Timing Diagram Laser On/Off Delay</u>



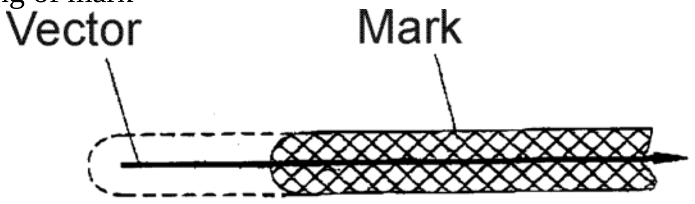
Laser On Delay (µsec)

- Generated at the beginning of a marking vector
- Keeps laser **off** until galvos have a chance to respond to command signals

Laser on delay too short:
- Will turn on laser before galvos reach set speed
- Will "dig" into part more
- "Pulse pile up"



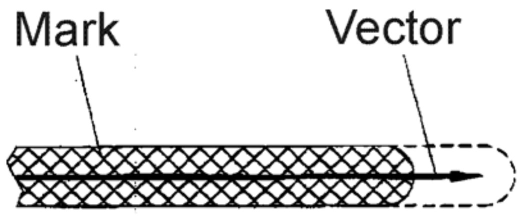
Laser on delay too long:
- Galvos reach set speed before laser is turned on
- Will miss beginning of mark



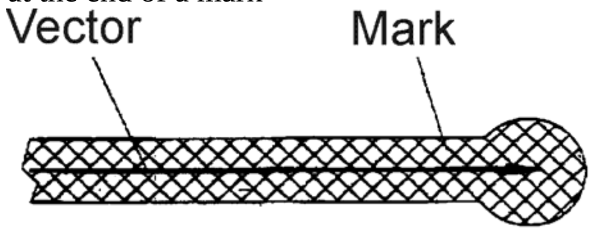
Laser Off Delay (µsec)

- Generated at the end of a marking vector
- Keeps laser **on** until end of vector reached

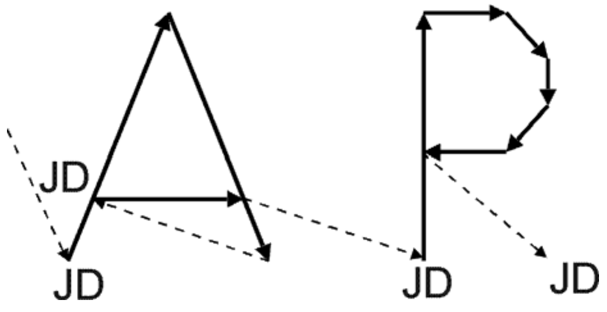
Laser off delay too short:
- Laser will be turned off before it reaches the end of a stroke
- Will miss end of mark



Laser off delay too long:
- Laser will stay on after it has reached the end of a stroke
- Can cause "blooming" at the end of a mark



Jump Delay (μsec)

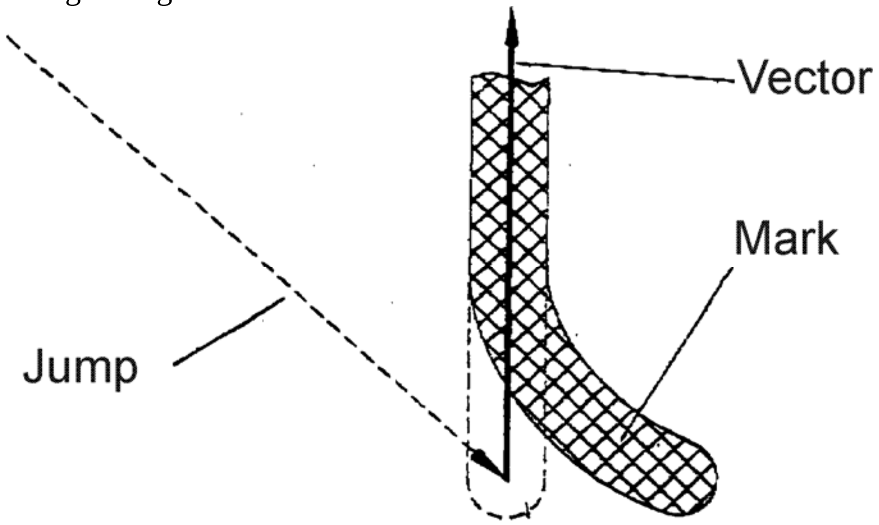


Jump Delay Description

- Generated at the end of a non-marking move (jump)
- "Settling time"
- Allows galvos to settle before a marking vector is started
- Necessary because jumps are usually faster than strokes, so galvos need more time to settle
- Laser is **off**

Jump Delay Too Short

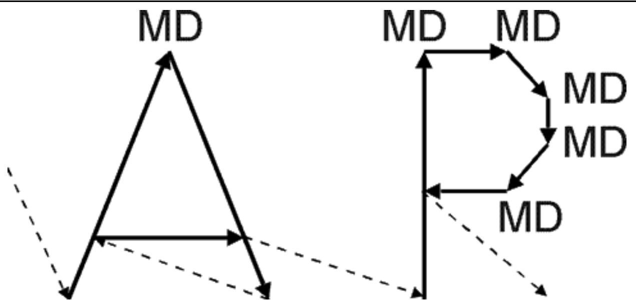
- Not enough time For galvos to settle properly
- Will start marking during "overshoot"



Jump Delay Too Long

- No visible effect on mark
- Will increase marking time

Mark Delay (µsec)



Mark Delay Description

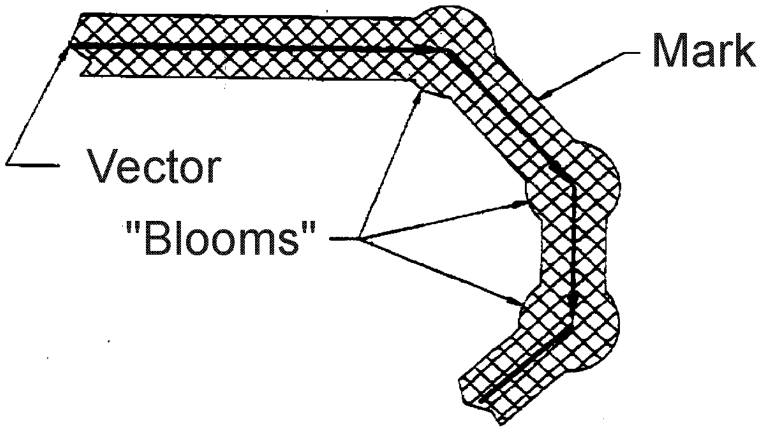
- Generated within a stroke (series of connected marking vectors)
- To maintain control of galvos when marking
- Typically a slight delay before changing direction
- Laser **not** turned off

Mark Delay Too Short

- Not much noticeable effect at normal marking speeds
- Some distortion possible at very high scanning speeds

Mark Delay Too Long

- Can cause line width to "bloom" on low threshold materials

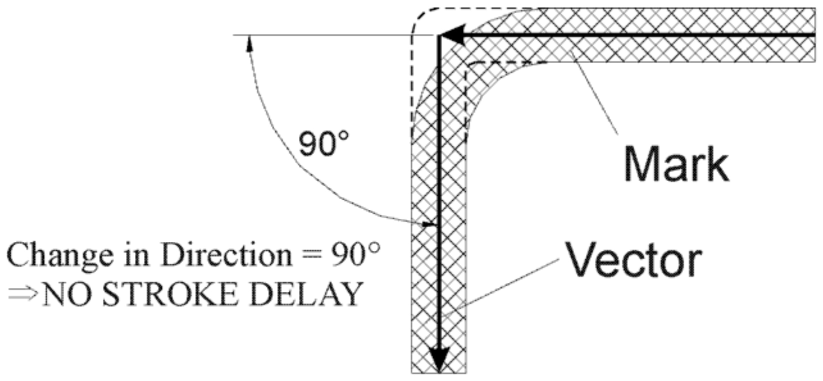


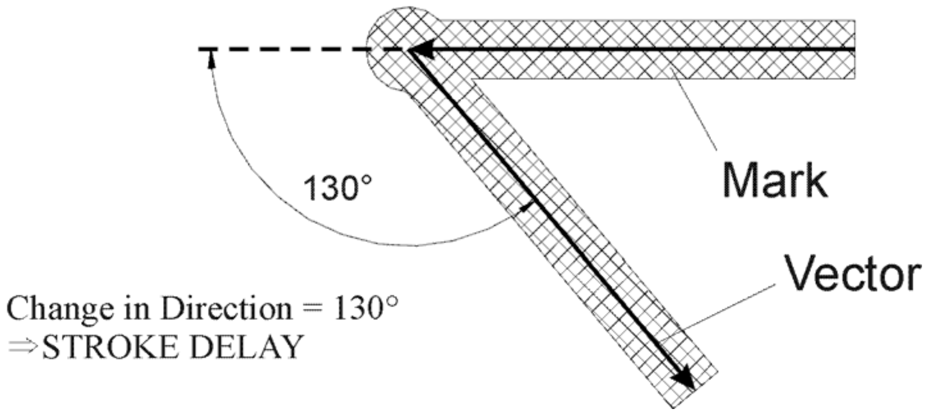
- Will increase mark time

Break Angle (degree)

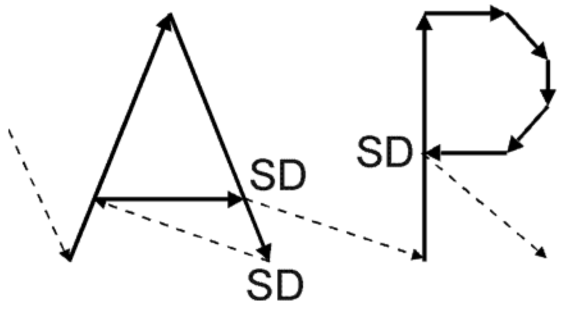
- To control galvos when turning corners at high speeds
- Inertia prevents immediate response to change in direction
- Sets the angle which activates break angle when exceeded
- Brakes one stroke in two separate strokes and inserts a stroke delay
- Disabled by setting Break angle = 180 degrees (off)
- The referenced angle is <u>not</u> the included angle but the change in direction.

Example:
Break angle set to 100°.
(This means the stroke is broken only if the change in direction exceeds 100°.)





Stroke Delay (µsec)



Stroke Delay Description

- Generated at the end of a stroke (series of connected marking vectors) before the next stroke or jump
- Allows beam to catch up with command signal and finish one stroke before continuing the next
- One stroke can be broken in two strokes if the included angle exceeds the Break Angle setting (see there)
- Need to balance with Laser Off delay

Stroke Delay and Laser Off Delay Interdependency

- Optimize Stroke Delay last because it depends on the Laser On and Off Delay
- Use large values for Stroke Delay initially, then slowly reduce
- Shortest possible Stroke Delay (SD):

$$SD \geq LOFF - LON - SP$$

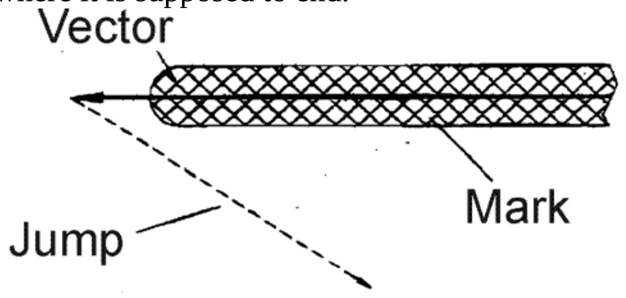
LOFF	=	Laser Off Delay
LON	=	Laser On Delay
SP	=	Step Period

Stroke Delay Too Short

Will not allow beam to reach the end of a stroke before sending the command to jump elsewhere.

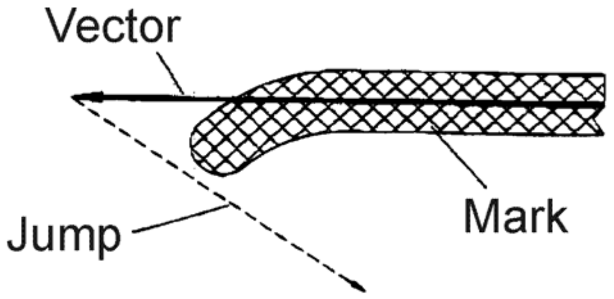
Stroke delay too short and laser off delay too short

If the Stroke Delay is shorter than the Laser Off Delay large parts of a vector can get lost.
Beam starts next move before it has caught up with the last move. Because the laser is turned off too early the mark ends before where it is supposed to end.



Stroke delay too short and laser off delay too long

Beam starts next move before it has caught up with the last move. Because the laser is still on this is visible.



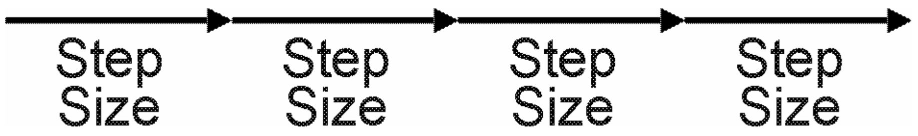
Stroke Delay Too Long

- Generally no visible effect
- Excessively long delays can cause abnormal software operation

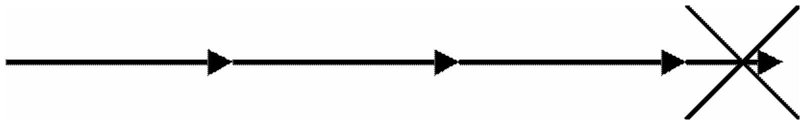
Autosegmentation

Micro-Vectors:

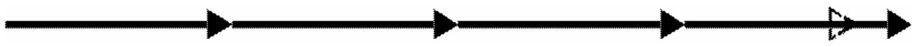
- Eliminates vectors shorter than half of the mark step size



Last Vector < Step Size/2:



Last Vector > Step Size/2:



Laser Power / Laser Power Delay

Laser Power (LSBs)

- Determines laser output power
- Digital signal from the **HC/2**
- Can be converted to an analog signal with the **I/O-2** add-on card

Laser Power Delay (msec)

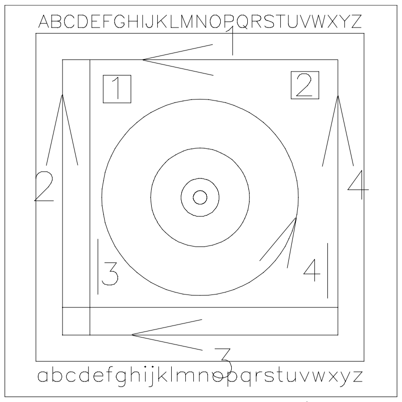
- Time delay which allows the power supply to reach a new setting before starting a mark
- Activated when changing between pens with different power settings
- **Power changes are not recommended within a mark because of the amount of time added to the mark. Power changes can also affect the laser life and power stability**

Lamp Current Delay Too Short

- Laser power will be changing at the beginning of the mark with a new pen
- If changing from higher power to lower power, the mark will dig too deep at the beginning of the mark
- If changing from lower power to higher power, the mark will start off weak of non-existent and fade into spec

Optimizing delays

Follow the steps described below for a quick way to optimized parameters. Use the pattern a similar pattern to the one below or generate your own test pattern. Mark the pattern at a size that is typical for your application.



General Scanning Test Pattern "$DELAYS.mcl"

Define Process Scan Velocity

Determine the scan velocity required for the particular process (material/laser dependent). For this we should use:

- Jump Size small
- Mark Delay = 0
- Break Angle = 90
- Stroke Delay large (>10000)
- Jump Delay large (>10000)

Mark straight lines and ignore the start-points and end-points of vectors.

Optimize Timing Between Laser and Galvos

For the process velocity now determined we have to optimize the timing between the laser and the galvos next.

- Mark a test pattern that clearly shows the start and end point of vectors.
- Find the LON and LOFF values that give you the best results:
  Full length of vector w/o blooming.

Optimize Jump Size and Jump Delay

Now we can start to reduce the jump-delay and increase the jump size until we start to see wiggle lines at the beginning of vectors **after** a jump.
The best ratio between jump speed and jump delay for the shortest cycle time will later depend on the number of jumps in the pattern which is actually to be processed.

Optimize Mark Delay

Increase the Mark Delay until we see no further improvement to small features, in particular small characters.

Optimize Stroke Delay

Finally we can reduce the stroke-delay. Beware of the limitations! Because of delay interdependencies we cannot go any smaller than:
$SD \geq LOFF - LON - SP$
Depending on your specific hardware (laser, galvo, mirror) it might not be possible to reduce the Stroke Delay to its theoretical minimum.

Laser Modulation

Laser Modulation Signal

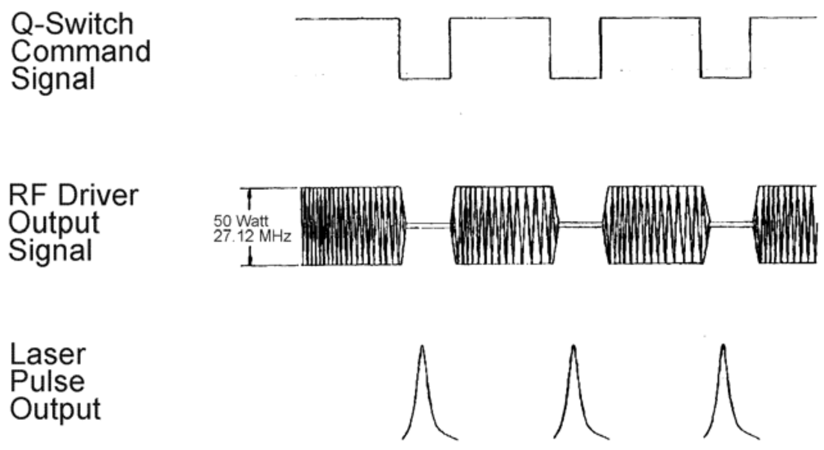
- Turns laser on and off
- Generated by the software
- Send out from **HC/2** or **HC/3** to the laser interface

Q-Switched Lasers (YAG)

- Pulses are produced by turning Q-switch on and off
- The laser pulse shape is affected by two things:
  1. How many times the Q-switch is turned off and on a second (referred to as frequency or Q-rate)
  2. How long the Q-switch is turned off for each pulse (Q-switch off time or pulse width)

Changing the Q-switch settings will affect the peak energy and average power of each pulse and is used to determine how the laser will interact with the material to be marked.

Timing Diagram



Rate and Power

Range:
- 0.02 KHz to 50KHz or CW
- Software selectable

CW (Continuous Wave):
- No Q-switching, RF power off
- No peak power, only average power

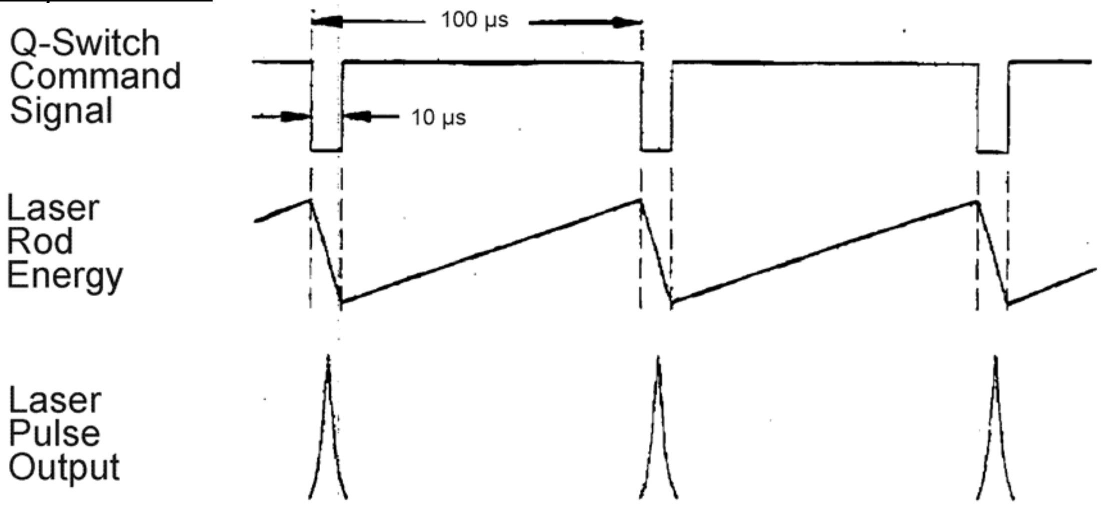
Low Q-Rates:
- Set in software by making the pulse width (off time) longer than the selected period (1/f)
- Higher peak power, lower average power
- Used more for "drilling" type of marking where high peak power is necessary to remove material

High Q-Rates:
- Lower peak power, higher peak power

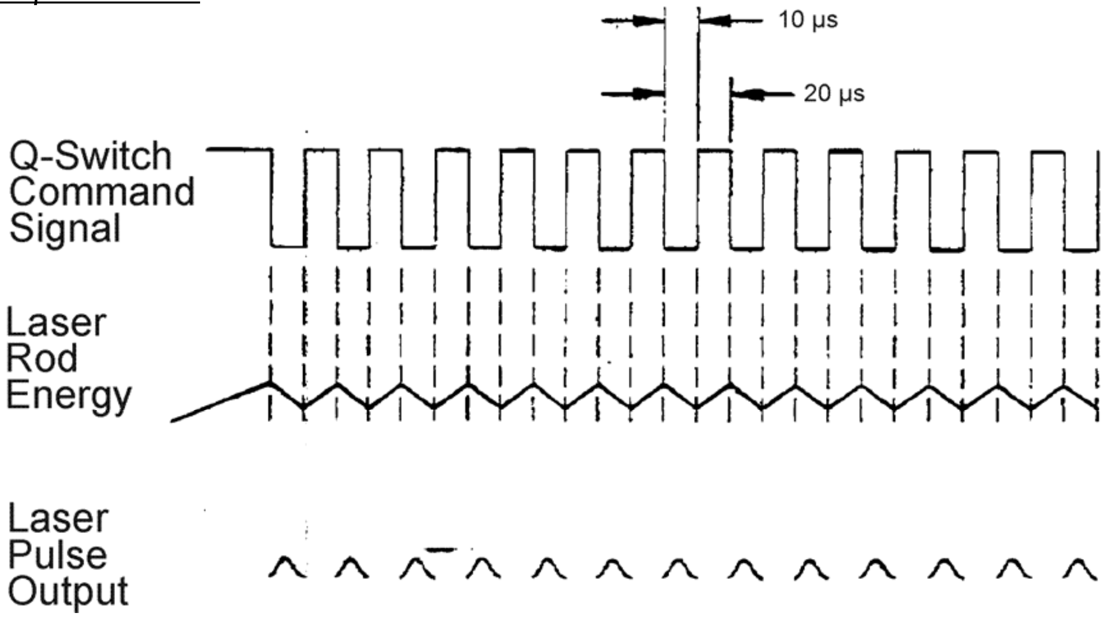
- Used more for a "heating" type of interaction with the material where the higher average power heats it up to discolor or burn in the mark

Q-Rate Variations

10 kHz / 10 µs Off Time:



50 kHz / 10 µs Off Time:



INDEX

END OF DOCUMENT